UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Máster en Ingeniería de Telecomunicación

TRABAJO FIN DE MÁSTER

Desarrollo de un prototipo de sistema electro-localizador en entornos acuáticos

Víctor Hugo García García

Tutor: Francisco de Borja Rodríguez Ortíz

JULIO 2016

Desarrollo de un prototipo de sistema electro-localizador en entornos acuáticos

Víctor Hugo García García

Tutor: Francisco de Borja Rodríguez Ortíz

Grupo de Neurocomputación Biológica (GNB)
Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
28049 Cantoblanco, Madrid
España

Julio 2016

Resumen

En entornos acuáticos ruidosos o fangosos, océanos profundos u otras zonas sumergidas donde algunos sistemas de detección del entorno habituales como las cámaras de video o el sonar pueden no resultar efectivos surge la necesidad de buscar una alternativa que permita reemplazarlos de manera efectiva y fiable. Para solventar esta necesidad, se ha tomado como referencia e inspiración una de las maneras en las que lo resuelve la naturaleza.

La evolución ha hecho desarrollar a una serie de especies un sentido muy especial y extraño para el ser humano: el sentido eléctrico. Estos animales, mayoritariamente acuáticos, poseen un sistema muy sofisticado que les permite detectar e incluso en algunos casos generar pequeños pulsos eléctricos y utilizarlos tanto para detectar objetos en sus inmediaciones ("electro-localización") como para interactuar con otros individuos ("electro-comunicación").

Inspirado en los peces eléctricos, en el laboratorio del Grupo de Neurocomputación Biológica se ha diseñado y construido un sistema robótico de experimentación con el objetivo de analizar y reproducir el proceso de electro-localización de manera replicable y precisa. Por este motivo, todo el sistema está desarrollado y diseñado con herramientas de código libre y tecnologías de impresión 3D, para poder ser fácilmente reproducido.

El proyecto, al que se le ha dado el nombre de "AquaRide", consiste en una plataforma robótica propia que implementa el esquema de movimiento CoreXY y que posee un sensor eléctrico que es capaz de desplazarse por un acuario detectando obstáculos a su alrededor utilizando exclusivamente las fluctuaciones del campo eléctrico emitido por el sensor.

AquaRide tiene como objetivo ser el sistema que abra las puertas a la realización de un robot de movimiento autónomo que incorpore un sensor de proximidad eléctrico como el desarrollado y esto le permita realizar una función de exploración en entornos no controlados.

A lo largo de esta memoria se explica cómo se han realizado los diseños de las piezas que componen el proyecto, el desarrollo del firmware para la placa Sanguinololu que controla el sistema y los programas en lenguaje Python que permiten el análisis de resultados.

Por último, se analizan los experimentos realizados para comprobar el comportamiento del dipolo ante objetos, mostrándose la capacidad del sistema de detectar y reaccionar ante elementos resistivos existentes en su trayectoria.

Palabras clave

Electro-localización, sentido eléctrico, AquaRide, sensado activo, subacuático, CNC, Robótica, Arduino, Sanguinololu, Gnathonemus Petersii, Diseño 3D, Python, CoreXY.

Abstract

In noisy or muddy underwater environments, deep waters or other submerged zones where some usual environment detection systems such as video cameras or sonars can be useless, a necessity for an alternative solution that can replace those in an effective and trustworthy way arises. To solve this need, the way nature solves this problem has been taken as a reference.

Evolution has developed in some species a very special and strange sense from the human species point of view; the electrical sense. These animals, mostly aquatic, posess a very sophisticated system that allows them to detect, (and even in some cases generate) small electrical pulses and use them both to detect objects nearby (electrolocation) and to interact with other individuals (electrocommunication).

Inspired by electrical fish, in the Grupo de Neurocomputación Biológica a robotic experimentation system was designed with the objective of analyzing and reproducing the process of electrolocation in a replicable and precise way. For this reason, the whole system was developed and designed with opensource tools and 3D printing techologies, increasing this way the reproducibility.

The project, named "AquaRide", consists of a robotic platform of our own design that implements the CoreXY motion technique, and has an electrical sensor that allows it to move through a fish tank detecting obstacles on its surroundings using exclusively the fluctuations of the electric field emitted by the sensor.

AquaRide wants to be the system that allows for the creation of an autonomous movement robot that incorporates an electrical proximity sensor similar to the one developed that allows it to explore non-controlled environments.

Throughout this document one can find explanations for the designs of the different 3D models that were created for this project, the development of the firmware for the Sanguinololu board that controls the system, and also the Python programs that perform the results analysis.

Finally, the results of the experiments are analyzed to check the behavior of the sensor, and the ability of detecting and reacting to resistive objects located in its path.

Keywords

Electro-location, electrosense, active sensing, AquaRide, underwater, CNC, Robotics, Arduino, Sanguinololu, Gnathonemus Petersii, 3D Design, Python, CoreXY.

Agradecimientos

Tras esta época que parece alcanza su fin, quiero agradecer vuestra presencia. Y lo quiero hacer dejando claro por qué repetiría cada uno de los días que he pasado entre vosotros.

En primer lugar, a Carlos, por su inagotable paciencia y buen humor, que han convertido tantas horas de trabajo en un divertido paseo.

También a, Ángel, Irene, Aarón y un intermitente Alex por convertir un laboratorio más en nuestro laboratorio. Guardaré muchos momentos vividos en el B-208, incluso con mi mala memoria.

Gracias a las semanas sin fin patrocinadas por Ana, que han conseguido que acabe el trabajo y aprenda algunas curiosas lecciones acerca de la vida.

A Paco, por su constante guía y consejo, y al resto de miembros del GNB gracias a los que nunca he tenido que pedir nada dos veces.

Al lector quiero agradecerle que se aventure a aprender sobre la electro-localización, y le animo a continuar el trabajo donde yo lo dejo.

Y por último, a mi familia y a mis padres que han conseguido no sin esfuerzo convertirme en quien soy hoy.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Glosario

IPI: Inter-Pulses Interval o Intervalo entre pulsos.

CNC: Computer Numeric Control. Se refiere a las máquinas cuya posición está controlada digitalmente

CoreXY: Técnica de movimiento basada en la hibridación de los ejes X e Y

Sensado activo: En el ámbito de la electrolocalización se refiere al sensado del medio utilizando impulsos generados por uno mismo

Motor paso a paso/ stepper motor: Dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos.

ADC: Conversor de señales de analógico a digital.

Sanguinololu: Sanguinololu es una solución electrónica todo en uno, de bajo costo para el control de dispositivos CNC.

Electrodo: Extremo de un conductor en contacto con un medio, al que lleva o del que recibe una corriente eléctrica.

CSV: Fichero de datos en el que cada valor está separado por comas u otro caracter preestablecido

CAD: Software de ayuda al diseño utilizado para crear dibujos o ilustraciones técnicas de precisión.

EDO: Elecric Discharge Organ. Órgano que permite emitir impulsos eléctricos.

ROV: Remotely Operated Vehicle.

1 Introducción

En la actualidad, los vehículos de exploración submarina (ROV) controlan su movimiento y posición utilizando principalmente cámaras de vídeo y sonar. Estos sistemas no obstante quedan comprometidos ante situaciones especiales como océanos profundos y entornos acuáticos turbios donde la iluminación sea un problema o lugares estrechos en los que las reverberaciones del sonar hagan imposible su uso. Es necesario por tanto encontrar un tipo de sensor que permita, junto a estos métodos, conformar un sistema sensorial integrado con los más habituales para ser funcional en estos entornos.

Figura 1.1: ROV del barco de investigación ártica Sir David Attenborough. Fuente de la imagen: NERC[30]

La naturaleza siempre va un paso por delante, y en los últimos años se ha descubierto que ya existe entre diversas especies de animales una solución alternativa muy interesante: El sentido eléctrico [1] [5]. Estos animales, mayoritariamente acuáticos, poseen un sistema muy sofisticado que les permite detectar e incluso en algunos casos generar pequeños pulsos eléctricos y

utilizarlos tanto para detectar objetos en sus inmediaciones ("electro-localización") como para interactuar con otros individuos ("electro-comunicación").

En concreto, en el Grupo de Neurocomputación Biológica de la Universidad Autónoma de Madrid, se ha investigado utilizando una de las especies animales más relacionadas con el sentido eléctrico: Los peces elefante o Gnathonemus Petersii (referencias de la [34] a la [40]).

Los peces elefante, sobreviven en medios de reducida visibilidad gracias a su capacidad para producir, recibir e interpretar pequeños impulsos eléctricos que utilizan, junto con el resto de sus sentidos [33] para la búsqueda de alimento, el reconocimiento del medio y la navegación así como la comunicación con otros miembros de su especie, con los que intercambian protocolos de cortejo o defensas territoriales, aunque todavía es desconocido cómo se realizan exactamente estos procesos de comunicación.

En la cola del Gnathonemus petersii (ver Figura 2.2) se sitúa el órgano encargado de producir estos impulsos eléctricos, en forma de pequeñas descargas. Este órgano se compone de miles de células eléctricas que se sitúan sobre la piel. Poseen además electro-receptores situados por prácticamente todo el cuerpo y éstos les permiten recibir los impulsos generados por otros peces, o los suyos propios emitidos con la función de comunicarse o detectar objetos u otros animales. Por tanto este órgano eléctrico es vital para esta especie, dado que sirve para orientación y para la búsqueda de alimento pero también para propósitos sociales. Estos incluyen la localización de sus congéneres, la ubicación de su posición jerárquica en el grupo y la obtención de pareja [1].

Los objetos que están dentro del campo eléctrico que generan estos peces suelen poseer una conductividad distinta a la del medio, y por tanto son detectados al alterar la corriente inducida a los órganos electro-receptores. El pez detecta estos cambios y la información que éstos llevan y a partir de ellos genera una imagen eléctrica del entorno que le rodea, la llamada electro-localización (ver Figura 2.3) [2].

1.1 Motivación y Objetivos

El proceso de la electro-localización, ha llamado la atención de biólogos, científicos e ingenieros, ya que supone un reto y a la vez una solución a ciertos problemas existentes, como el introducido al principio de este texto: la navegación submarina en entornos desfavorables.

En este tipo de entornos, donde se ha de realizar una reconstrucción de las inmediaciones en espacios submarinos estrechos o fangosos es exactamente donde el sentido eléctrico brilla por su eficacia.

Tras toparse la ciencia con este método de localización, diversos estudios se han realizado en este área, pero en general se han utilizado sistemas que no facilitan la réplica de los experimentos[4] [19][21]. Por este motivo una de las directrices que tiene este Trabajo de Fin de Máster es desarrollar un sistema fácilmente repetible, que permita adentrarse en el área de la electro-localización sin que esto suponga el desarrollo de un sistema de experimentación nuevo, sino que se pueda reproducir este mismo sistema en cualquier otro laboratorio. Esto obliga al sistema a estar completamente desarrollado utilizando herramientas de distribución libre, como OpenSCAD o Python.

A consecuencia de lo anterior, la segunda directriz ha de ser necesariamente que el proyecto sea fácilmente mejorable y actualizable, ya que cualquier experimento o proyecto que se desee basar en AquaRide deberá tener una base sólida sobre la que poder asentarse.

Por tanto, el proyecto está enfocado al uso de un esquema que ofrezca :

- La mayor cantidad de información posible con el sistema más simple posible.
- La capacidad de aumentar la complejidad del sistema fácilmente en cualquier dirección, ya sea mecánicamente o en la parte de software, implementando nuevos algoritmos de manera sencilla.
- Facilidad en la implementación de partes de AquaRide en otros proyectos similares o más avanzados, posibilitando así un avance mucho más rápido en la investigación sobre la electro-localización.

En cuanto al núcleo de este proyecto, se pretende desarrollar una plataforma robótica bioinspirada, equipada con "sentido eléctrico" similar al que disponen los peces eléctricos en la naturaleza. Para crear un entorno en el que poder recrear esta capacidad:

- En una primera fase se diseñara un sensor eléctrico sumergible bioinspirado.
- En una segunda fase se realizará una adaptación de un acuario para incluir un sistema mecánico que pueda mover digitalmente este sensor, basándose en ciertos modelos de electro-localización.

- Por último, se conformará el sistema completo y se validará comprobando que el sensor sea capaz de modificar sus propios desplazamientos en base a los elementos detectados a través de las medidas de campo eléctrico obtenidas.

En cuanto a los hitos de desarrollo de este proyecto, se pueden ver en la siguiente lista:

- **Objetivo Nº1**: Diseño de un sistema mecánico capaz de soportar y desplazar por un acuario una sonda electro-receptora.
- **Objetivo Nº2**: Construcción de dicho sistema y adaptación al acuario.
- **Objetivo Nº3**: Instalación en un ordenador del software y preparación hardware necesarios para el control de la posición del sistema mecánico en el acuario.
- **Objetivo Nº4**: Diseño y construcción de una sonda electro-receptora que permita ser modificable y personalizable para distintos modelos de electro-localización.
- **Objetivo Nº5**: Conexión del sistema DAQ y el de procesado de datos.
- **Objetivo Nº6**: Validación del sistema completo mediante la realización de experimentos:
  - Calibración y validación del sistema de desplazamiento CNC.
  - Validación de la sonda eléctrica.
  - Validación de la fase de procesamiento de los datos obtenidos.
- **Objetivo Nº7**: Modificar el movimiento de la sonda en base a la información recogida por la misma durante el propio experimiento, es decir, en lazo cerrado.

1.2 Metodología de trabajo

Todo el proyecto se ha desarrollado en varios ordenadores distintos, con sistemas operativos Windows 10 y Ubuntu 12.04 y 14.04.

En cuanto al software utilizado toda la información del proyecto, tanto datos como código y programas, han sido manejados utilizando la plataforma “Github”, que permite alojar proyectos utilizando el sistema de control de versiones Git.

Las herramientas de diseño y desarrollo han sido:

- OpenSCAD para el desarrollo de piezas en 3D.
- Arduino IDE para la creación del firmware para la placa Sanguinololu.
- gedit como editor de texto enriquecido para el desarrollo del código Python.
- LibreOffice para la documentación y la escritura de la memoria.

Como se puede observar, todos los programas son de código abierto y multiplataforma y permiten colaborar en el proyecto desde cualquier lugar, ordenador y sin ningún coste.

1.3 Organización de la memoria

El cuerpo de la memoria consta de los siguientes capítulos:

1. Introducción.
2. Estado del arte. En esta sección se aborda el conocimiento existente actualmente sobre los principales temas e investigaciones realizadas en las que se basa el proyecto.
3. Diseño e implementación. Aquí se desarrolla de manera más pormenorizada el proceso de diseño y construcción, así como los problemas encontrados y las soluciones utilizadas.
4. Experimentos. En esta sección se abordan los experimentos realizados una vez se ha puesto en marcha el sistema y cómo se ha realizado la detección de objetos.
5. Conclusiones y trabajo futuro. En este último capítulo se resume el resultado del trabajo, con una lista de los objetivos alcanzados. También se comenta cuales son las pautas naturales a seguir en el proyecto, así como las conclusiones extraídas del trabajo realizado.
6. Anexos. En esta parte de la memoria se adjuntan documentos adicionales que no tienen cabida dentro de la memoria pero relevantes para el proyecto, como el código de los programas desarrollados o manuales de puesta en marcha.

2 Estado del arte

Dada la naturaleza multidisciplinar de este trabajo, el estado del arte engloba diversas áreas de trabajo generalmente no relacionadas. En esta sección se va a hacer una panorámica para situar al lector en los siguientes campos:

- El fenómeno de la electro-localización y electro-comunicación aplicado a los peces eléctricos, más concretamente al pez elefante.
- Los modelos eléctricos existentes para sondas electro-receptoras: el proyecto ANGELS
- La electrónica de adquisición de datos.
- Sistemas de control numérico.

2.1 El fenómeno de la electro-recepción y la electro-comunicación aplicado a los peces eléctricos

La capacidad de ciertos animales para detectar y utilizar en su favor los campos eléctricos ha sido durante siglos un foco de interés para biólogos y científicos[1]. En la naturaleza este sentido eléctrico no ha aparecido de manera única, sino que existe en numerosos casos y, lo que es más interesante, a través de distintas ramas evolutivas. Esto da lugar a pensar que la electro-recepción es una capacidad muy beneficiosa y por tanto, interesante de cara a un estudio científico.

Debido a las propiedades conductoras que posee el agua frente al aire, la mayor parte de las especies con células electro-receptoras son acuáticas.

En el grupo más generalista y extenso, el de los electro-receptores, se engloba a todos los seres que no necesariamente son capaces de generar campos eléctricos pero que pueden sin embargo detectarlos y utilizar este sentido para actividades como la depredación. La actividad nerviosa y muscular de las presas genera sutiles cambios eléctricos en el medio que son interpretados por el depredador, a partir de los cuales puede realizar una identificación y posicionamiento de su presa. Son notables en este ámbito los tiburones, que poseen células receptoras capaces de detectar diferencias del orden de los nanovoltios en bajas frecuencias [2][7].

Un subgrupo más interesante todavía es el de aquellas especies que han desarrollado la capacidad de no sólo advertir cambios en los campos eléctricos circundantes, sino también de provocarlos activamente.

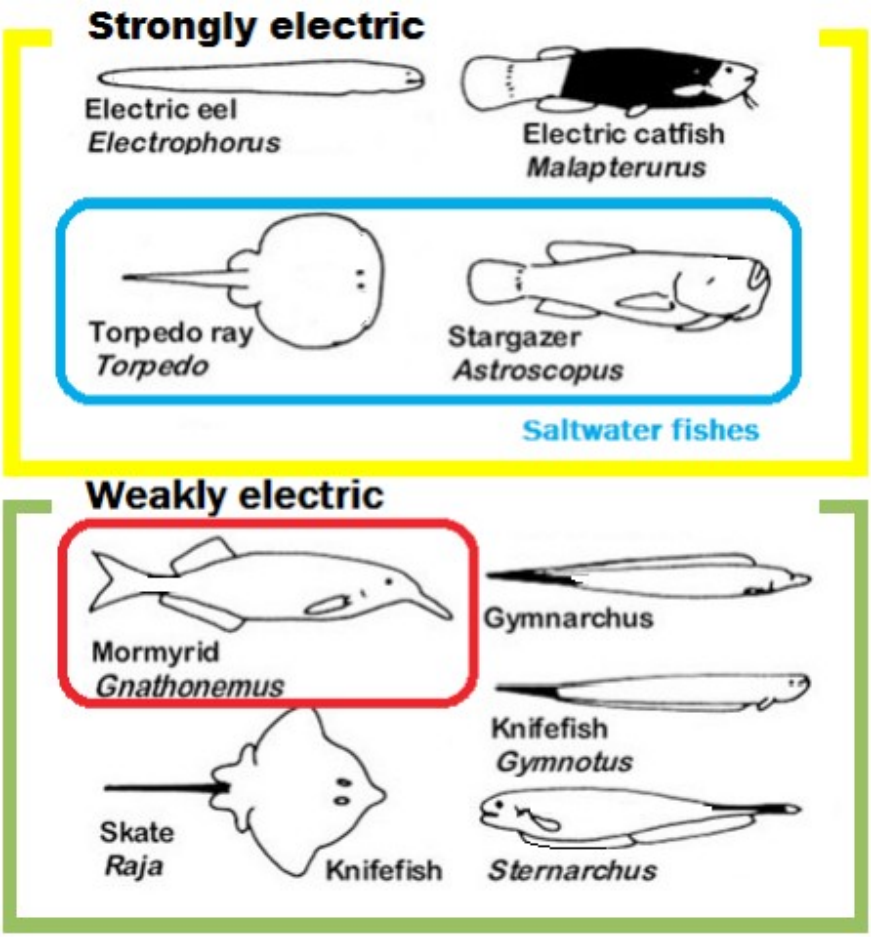


Figura 2.1: Desglose de los distintos peces eléctricos. Resaltado en rojo el género Mormyrid Gnathonemus al que pertenecen los Gnathonemus Petersii.

Este es el grupo de los peces eléctricos y se puede dividir en dos grandes grupos (Ver Figura 2.1):

- Los fuertemente eléctricos (como las anguilas eléctricas), capaces de generar descargas de más de 450 voltios y que utilizan esta habilidad fundamentalmente para la depredación o la defensa.
- Los débilmente eléctricos, que utilizan esta capacidad para la electro-comunicación y la electro-localización. Este tipo de peces son ideales para el estudio científico gracias a que por su naturaleza es fácil implementar interfaces de comunicación bidireccional. Además, dentro de este grupo existen otros dos subgrupos atendiendo al tipo de emisión eléctrica realizada. Algunos de estos peces emiten señales tipo pulsos de manera intermitente, mientras que otros envían señales tipo onda de manera continua. En este caso, el pez Elefante o Gnathonemus petersii que ha sido utilizado en las investigaciones realizadas en

el Grupo de Neurocomputación Biológica realiza emisiones del tipo "pulso". Más adelante se verá que esto es un parámetro que se puede emplear para realizar electro-localización artificial de distintas formas

2.2 Electro-localización

La principal característica de la electro-localización activa frente a la pasiva es la propia emisión de impulsos eléctricos para realizar la reconstrucción del medio. Este mecanismo es característico de los peces débilmente eléctricos como el pez elefante, al cual proporciona la posibilidad de navegar por aguas turbias o sin visibilidad, las cuales son típicas en su hábitat natural [3][6][13].

Todos los peces eléctricos poseen un conjunto de células eléctricas derivadas de células musculares o neuronales que forman el órgano eléctrico (EDO), encargado de emitir los impulsos eléctricos. En el caso de los Gnathonemus, dentro de los Mormíridos, este órgano esta formado por miles de células concentradas en la zona del pedúnculo caudal [8] (ver Figura 2.2).

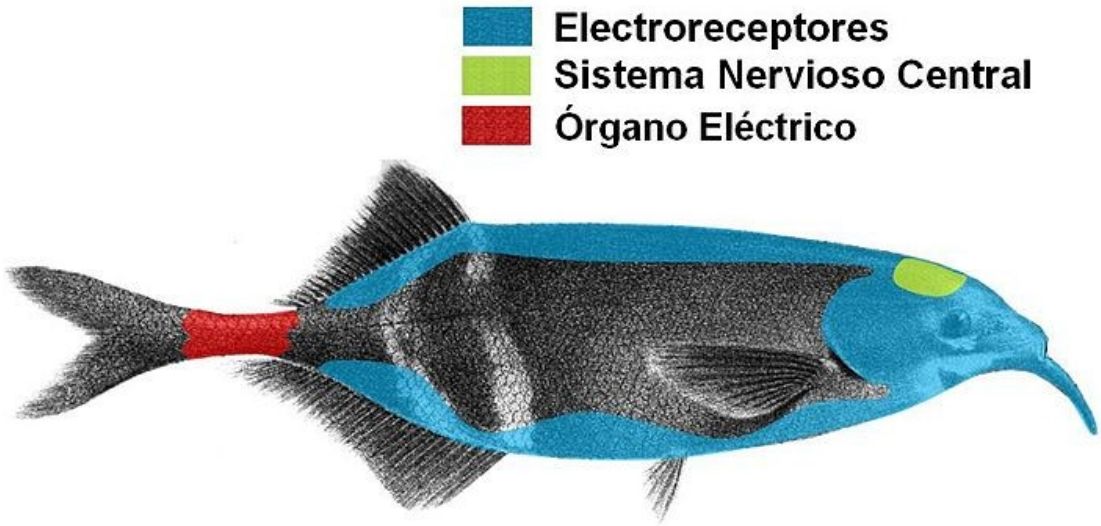


Figura 2.2: Localización del pedúnculo caudal, donde se encuentra el EDO (marcado en rojo), los electrorreceptores distribuidos por la piel y el sistema nervioso central del Gnathonemus_petersii.
Imagen creada por Juan Vera-Gueico y tomada de: https://commons.wikimedia.org/wiki/File:Gnathonemus_petersii_%28G%C3%BCnther,_1862%29.JPG

Aquellos objetos cercanos con una resistencia o conductividad distinta a la del agua, perturban el campo que percibe el pez (ver Figura 2.3). Estas perturbaciones en la conductividad del medio son recibidas por las células electro-receptoras situadas por toda la superficie del cuerpo del pez. El pez elefante entonces interpreta y procesa esta información, generando entonces la llamada imagen eléctrica [9][13][14]. Tras crearse una imagen eléctrica del entorno, el pez reacciona ante

lo que "siente". Este proceso ocurre una y otra vez durante toda la vida del pez, que emite pulsos a una velocidad media aproximada de 150 pulsos por minuto, reduciéndose en casos de extrema tranquilidad (aislamiento social y territorio muy conocido) y aumentando en caso contrario.

La precisión de los receptores eléctricos de los peces elefantes es tal que les permite además estimar el valor de resistividad o capacidad de un objeto a su alcance dependiendo del comportamiento del campo y las perturbaciones que produzcan en él los objetos a su alrededor.

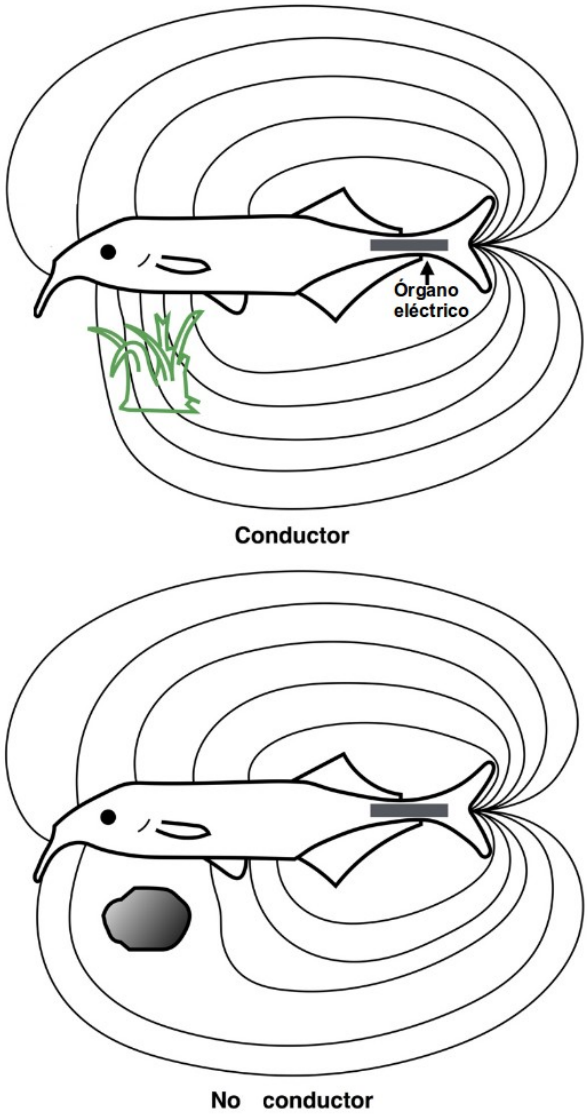


Figura 2.3: Electrolocalización. Las líneas de campo eléctrico emanan del órgano electrico situado en el pedúnculo caudal y son detectadas por las zonas de piel electror-receptivas. Estas se concentran alrededor de los elementos más conductivos que el agua y se dispersan alrededor de los menos conductivos[9]. En las figuras se puede ver la diferencia entre el efecto que tienen los elementos conductivos (verde) y resistivos (gris). Imagen modificada tomada de [31].

La capacidad de discriminación eléctrica de cada pez está relacionada directamente con el tamaño del pez, el tamaño del objeto y la diferencia en su conductividad con respecto a la del medio [11].

2.3 Electro-comunicación

El canal eléctrico es invisible para la mayoría de las especies y por este motivo, los peces que utilizan estos impulsos para la comunicación han tenido una importante ventaja evolutiva. Este "canal privado" de comunicación se basa en que un individuo emite impulsos muy similares a los utilizados para la electro-localización (de bajo potencial, pero con frecuencias de emisión más altas) y son recibidos por los electro-receptores de otro individuos, que los interpretan en función de la frecuencia de emisión (IPI – Inter-Pulses Interval) y la longitud del pulso (ver Figura 2.4). A partir de esta información pueden determinar aspectos como el sexo o el dominio territorial, así como procesos de cortejo o ataque [1][8][10][14].

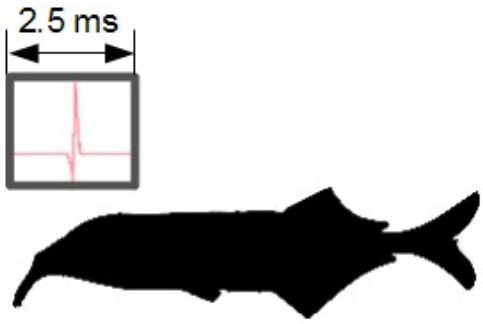


Figura 2.4: El pulso del Gnathonemus petersii.

Muchos estudios se han realizado sobre este área, tratando diversos temas relacionados como la capacidad de modulación de las señales, que se investiga con la intención de mejorar las áreas de procesado y compresión de la información, aprendiendo de los métodos utilizados en la naturaleza[12].

Sin embargo la electro-comunicación no queda relegada al campo de los seres acuáticos. Fuera del agua, las abejas productoras de miel poseen también receptores eléctricos y son capaces de detectar, por ejemplo, si otra abeja ha recolectado el polen de una determinada flor en base al potencial en sus pétalos, debido a que al volar de flor en flor las abejas se cargan de energía electrostática, que depositan al posarse [5].

2.4 La navegación submarina por sentido eléctrico.

La navegación submarina está actualmente basada esencialmente en el uso de la visión y el sonar. La primera de estas dos está restringida principalmente por la disponibilidad de luz en el ambiente, lo cual supone un problema a grandes profundidades o zonas de difícil acceso como cuevas.

Por el contrario, el sonar es un sistema de sensado activo, lo que quiere decir que la energía es proporcionada por el propio sensor. Esto permite trabajar en entornos más difíciles de reconocer,

y de hecho la eco-localización es empleada por algunos animales tanto acuáticos como terrestres, entre los cuales se encuentran por ejemplo los murciélagos, las orcas y los delfines.

No obstante, la eco-localización o biosonar también tiene algunos problemas. En entornos demasiado estrechos o confinados se pueden observar reverberaciones que dificultan enormemente la interpretación de las señales recogidas. Es en este tipo de entornos donde la electro-localización puede tener un papel importante en la reconstrucción del entorno, donde la integración de este sistema de reconstrucción con los habituales pueda aportar valiosa información [20][22][23][24]. Por este motivo, se han realizado ya diversas investigaciones dirigidas a la aplicación de la electro-localización en el ámbito de la robótica, como las desarrolladas por los departamentos de Ingeniería Mecánica y Ingeniería Biomédica en la Northwestern University en Evanston, Illinois, donde trabajan en, por ejemplo, la localización de objetos y estimación de la orientación a través del uso de campos eléctricos[17][18].

2.4.1 El proyecto ANGELS

La principal referencia que se ha tomado en este proyecto de cara al desarrollo de la plataforma es el proyecto "ANGuilliform robot with ELectric Sense (ANGELS)" [16][19][21]. Este proyecto tiene por objetivo el desarrollo de un robot autónomo capaz de desplazarse por un entorno subacuático guiándose exclusivamente por el sentido eléctrico, y cuenta con numerosas publicaciones y distintos avances tanto en distintos aspectos como son el filtrado de los datos recibidos como los modelos eléctricos empleados.

2.4.2 El modelo eléctrico

Tomando inspiración del mecanismo utilizado por los peces elefantes, en el proyecto ANGELS se ha diseñado un sensor de la manera que se expone a continuación.

La versión más simple de este sensor está formada por dos electrodos en los extremos de una estructura cilíndrica. A estos electrodos se les impone una diferencia de potencial que genera un campo de tipo dipolar similar a la creada por los peces elefante gracias a su órgano eléctrico (Ver Figura 2.5).

El campo generado dependerá de la resistividad del medio y variará su forma dependiendo de si existen objetos que perturben esta resistividad. En este caso, la manera de obtener datos del medio es realizar una medida de la corriente que es capaz de atravesar este volumen alrededor del sensor y detectar las variaciones en la misma.

Este esquema es del tipo al que denominan como V-I (emisión de voltaje y medida de corriente a través de un amperímetro), y tiene la ventaja de que necesita un número más reducido de electrodos para hacer medidas más simples, ya que es suficiente con tener un electrodo de emisión y un electrodo por cada punto en el que se desee hacer una medida.
En las etapas iniciales del proyecto se propuso utilizar este modelo de medida, pero se descartó debido a los motivos que se exponen a continuación.

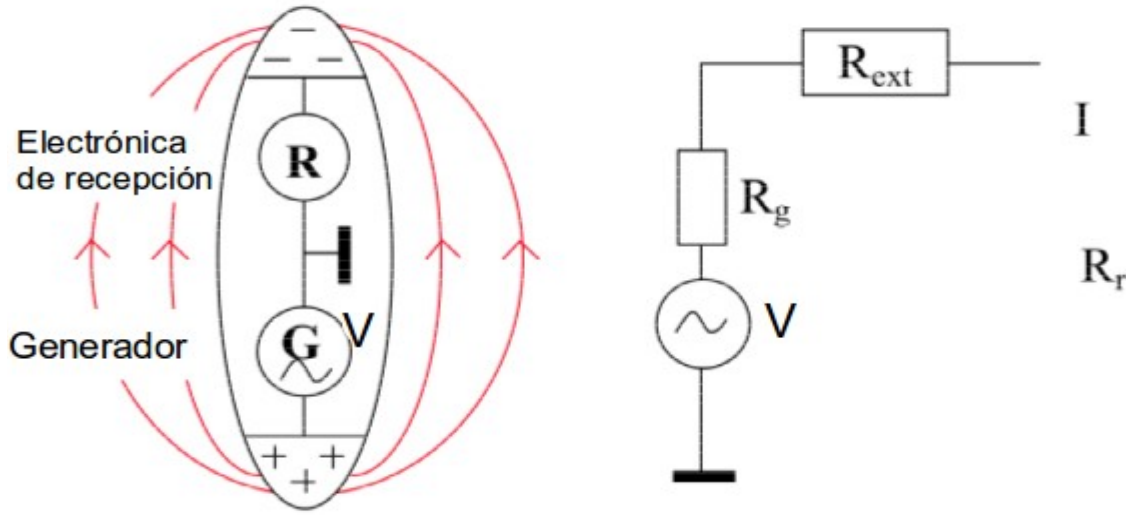


Figura 2.5:A la izquierda, el sensor simplificado tipo V-I compuesto de dos electrodos, utilizado en el proyecto ANGELS. A la derecha, el circuito simplificado del sensor, donde V es el generador de impulsos, Rg es la resistencia del generador, Rr es la resistencia de la electrónica de recepción del sistema y Rext es la resistencia del medio externo, es decir, el agua y los posibles objetos que rodeen a la sonda. Figura adaptada de [16].

1. El principal problema de este tipo de sensado es que sólo se puede establecer un bucle de corriente para cada medida, ya que al ser cada amperímetro un sumidero de corriente, dos medidas simultáneas no controladas pueden interferir entre ellas.
2. En el caso de que se tuviese en cuenta este fenómeno, el modelo se complicaría significativamente.
3. La solución mas sencilla a este problema por tanto sería alternar la medida de los distintos electrodos para una misma emisión de señal, pero este método sin embargo tiene una complicación, y es que se está añadiendo una dispersión temporal entre las medidas de los sensores y una dilatación muy significativa en los tiempos de medida, ya que si se utiliza por ejemplo un sensor con 40 electrodos, se necesitan 40 medidas individuales activando cada uno de

los amperímetros, mientras que con el modelo V-V se realizarían todas seguidas de manera prácticamente instantánea.

Debido a estos motivos y a la búsqueda de la opción funcional más sencilla, se ha optado por un modelo tipo V-V, cuyo funcionamiento y ventajas se explican en detalle en la sección 3.4.1: Esquema tipo V-V.

2.4.3 Sistemas de control numérico

Las Máquinas de control numérico (CNC por las siglas en inglés: Computer Numeric Control) [29] representan la adaptación de la era digital a las herramientas de posicionamiento. Estos sistemas son controlados generalmente a través de motores paso a paso o servomotores y representan un gran avance con respecto a las máquinas tradicionales, ya que además de su precisión superior es posible programarlas para fabricar distintos tipos de piezas rápidamente.

Este tipo de sistemas han sido y son fundamentalmente empleados en la industria para fresados, cortes y demás mecanizados de piezas. En general, estas máquinas son caras y específicas para cada tipo de aplicación, pero recientemente ha surgido un mercado de CNCs que presentan muchas ventajas para su aplicación en un proyecto como el que se presenta.

Se trata del mercado de las impresoras 3D. Este tipo de sistemas son una perfecta fuente de inspiración, ya que resuelven el problema de movimiento cartesiano de una manera sencilla, robusta y precisa [32]. Además, esta industria ha sido empujada en un inicio por una comunidad de desarrolladores libres que han apostado por difundir sus prototipos y sistemas.

Actualmente existen una increíble variedad de modelos tanto comerciales como de libre distribución y el mercado está en pleno auge, permitiendo que tanto los componentes para este tipo de máquinas como los diseños estén fácilmente al alcance de todo el mundo.

3 AquaRide: Diseño e Implementación

A continuación se detalla el proceso de diseño, la toma de decisiones en cuanto a métodos y herramientas a utilizar y también cómo se han llevado a la práctica las mismas, así como los problemas encontrados durante la implementación y sus soluciones.

3.1 El sistema completo

Antes de proceder a explicar cada una de las partes que componen AquaRide, en esta sección se va a aclarar en qué partes se subdivide el proyecto para proporcionar una noción general y facilitar el seguimiento del resto del capítulo. En cuanto al montaje exacto del circuito, con las conexiones específicas, está desarrollado en el anexo B: Conexiones y montaje".

AquaRide necesita para funcionar cuatro componentes esenciales:

- Parte Hardware.
  - La plataforma de desplazamiento (Ver Sección 3.2)
    - Se encarga de mover el sensor por la pecera y dar soporte a los cables del sensor.
    - También provee a la placa Sanguinololu de la información de fin de carrera para calibrar la posición.
  - La placa Sanguinololu (Ver Sección 3.3).
    - Sirve como interfaz entre el PC y el sistema de motores y el sensor.
    - Traduce los comandos del ordenador y los transforma en movimientos de los motores, operaciones de sensado y filtrado de datos.
    - Da formato a los datos obtenidos y los envía junto con la posición adquirida de vuelta al ordenador.
  - El sensor (Ver Sección 3.4)
    - Su labor es portar los electrodos.
- Parte firmware (Ver Sección 3.5)
  - Firmware implementado para la placa Sanguinololu.
    - Implementa los algoritmos que llevan a cabo las tareas requeridas para la placa.
- Parte software (Ver Sección 3.6)
  - El ordenador.
    - Ejecutará los programas que indiquen a la placa Sanguinololu los movimientos a realizar.

- Para cada dato obtenido, evalúa la posibilidad de encontrar objetos y determina cambios en el movimiento previsto del sensor.
- También es el encargado de realizar la representación de los datos obtenidos para cada experimento.

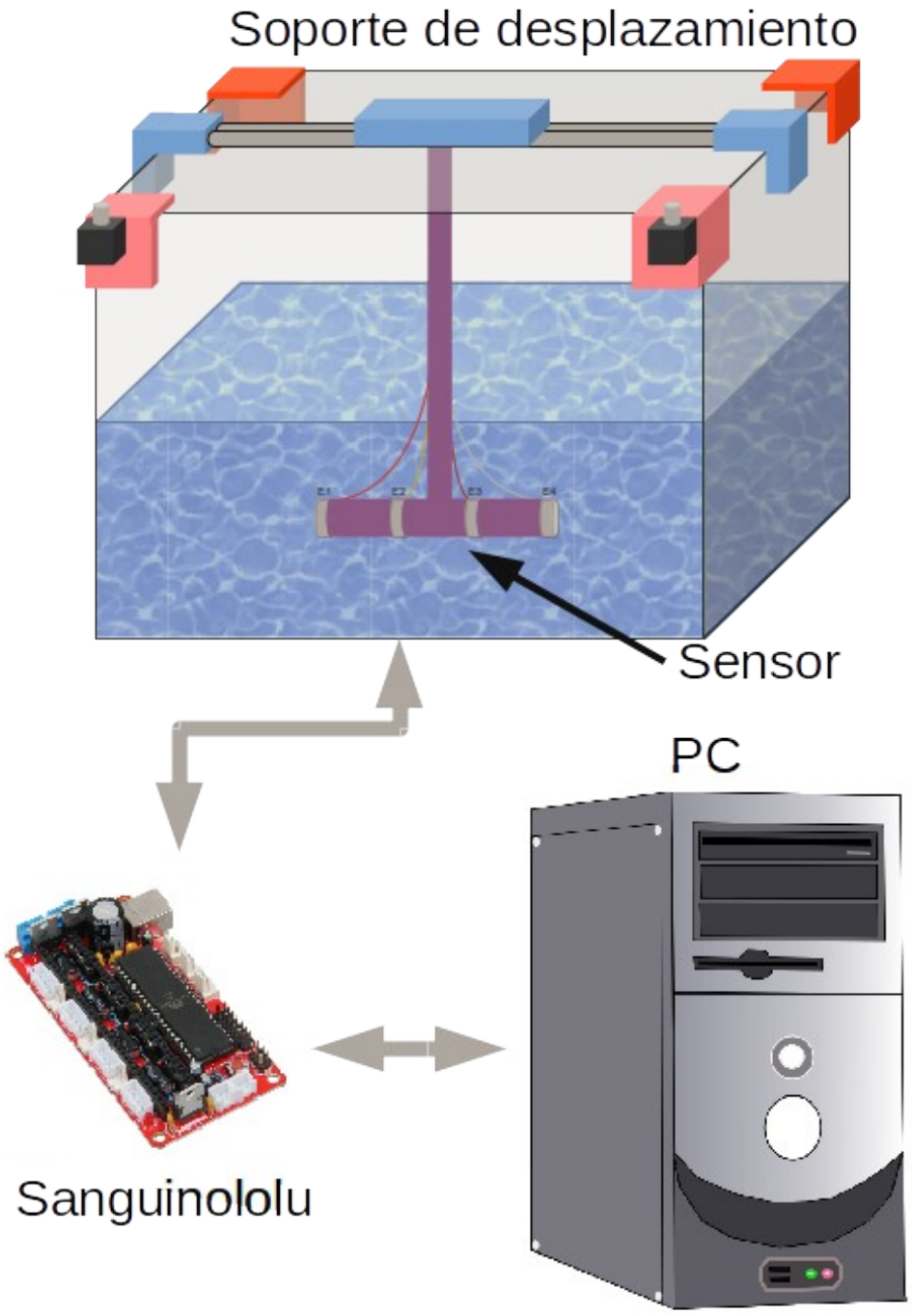


Figura 3.1: Esquema simplificado de AquaRide. Un diagrama completo con las conexiones detalladas se puede encontrar en el anexo B: Conexiones y montaje.

3.2 Hardware: CoreXY aplicado a un acuario.

Al comienzo del proyecto, la única directriz que se tenía acerca del resultado final era que se debía controlar la posición de una sonda dentro del acuario. Esto implica la primera decisión acerca del tipo de control que se va a usar. Existen dos alternativas muy diferentes:

- Utilizar un robot submarino con control autónomo de la posición.

El principal problema de esta opción es que el agua es un medio en el que la posición depende de varios factores poco predecibles como las corrientes, la flotabilidad del robot o la temperatura. Por otro lado, las pocas soluciones comerciales existentes en cuanto a robots de navegación submarina están muy enfocadas al ocio. No obstante, el objetivo final de este proyecto es la posibilidad de implementar el sentido de la electro-localización en un robot autónomo de este tipo.

- Utilizar una plataforma que controle su posición de manera externa, con un sistema de sujeción que permita evitar los efectos de corrientes y mantener una posición fija de manera sencilla.

Esta última fue la opción escogida, ya que aunque tiene las limitaciones de "Entorno de pruebas", permite un estudio del fenómeno de la electro-localización mucho más preciso, gracias a un control de la posición mucho más riguroso.

Al concluir que la solución más adecuada para el prototipo era el uso de una plataforma externa al acuario, el siguiente paso es decidir qué método se va a utilizar para la locomoción.

En este caso existe una infinidad de opciones, pero este proyecto se ha inspirado en la tecnología de impresión 3D ya que resuelve un problema muy similar de movimiento cartesiano. Este área además tiene una ventaja, y es que posee una comunidad de desarrolladores, código y diseños libres enfocados a hacer esta tecnología un producto fiable, preciso y sobretodo asequible para todo el mundo.

Las opciones en cuanto a la configuración de los motores y los ejes son abundantes, pero AquaRide presenta unas restricciones muy específicas y distintas a las que se presentan habitualmente en los sistemas de impresión 3D. Al construirse sobre un acuario, se necesita imponer los siguientes criterios:

- El diseño ha de ser flexible e implementable en distintos modelos de acuarios, independientemente de sus dimensiones y características.

- Desplazar en la medida de lo posible los componentes eléctricos lejos de la zona con riesgo de mojarse, para aumentar la seguridad del sistema
- AquaRide debe ser ágil. Esto es debido a que el proyecto ha de poder reaccionar a obstáculos encontrados en el camino y corregir su movimiento. Para permitir esto, se debe desplazar todo el peso posible de la parte móvil del sistema, minimizando así las inercias generadas.

Teniendo en cuenta todas estas características se ha elegido la configuración conocida como CoreXY[25]. A continuación se expone cómo funciona este tipo de configuración mecánica y cómo se ha adaptado al sistema motor de AquaRide.

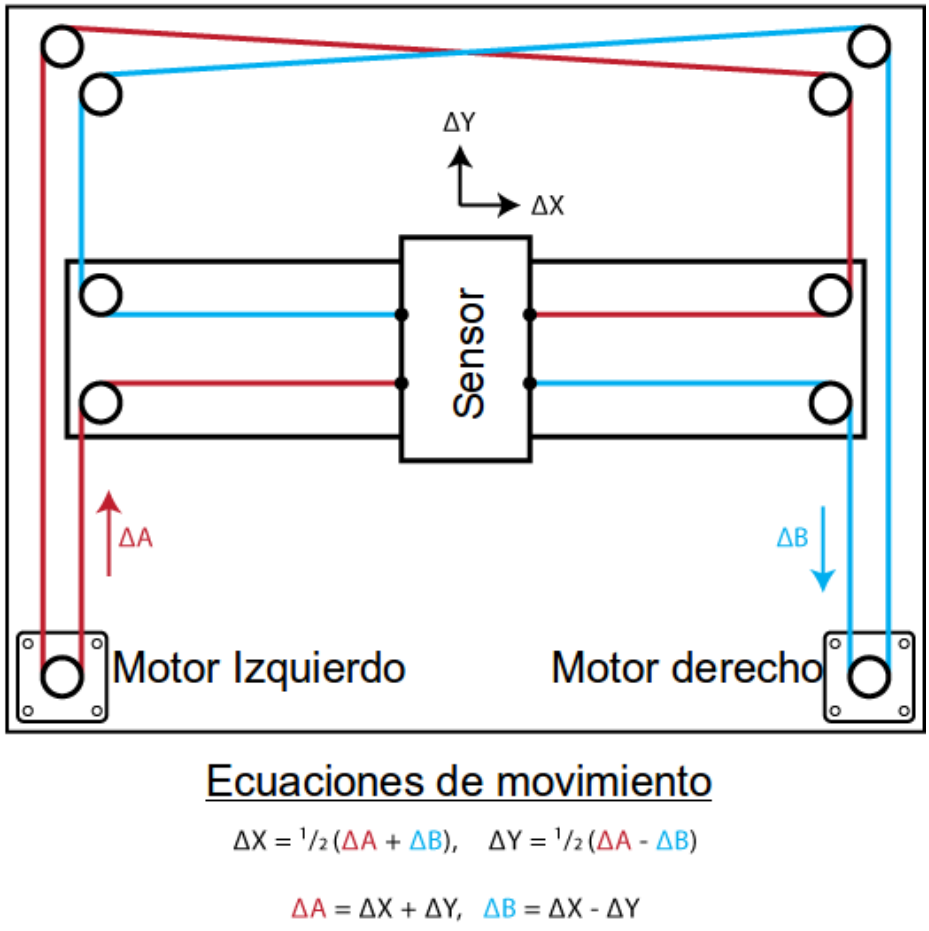


Figura 3.2: Esquema de funcionamiento del sistema CoreXY. Un movimiento en el eje X se logra moviendo ambos motores en la misma dirección, mientras que un movimiento vertical se consigue moviéndolos en direcciones opuestas. Figura adaptada extraída de: http://corexy.com

Tal y como se ve en la Figura 3.2, CoreXY es la manera de denominar a este esquema de movimientos, en el que no existe un "motor para el eje X" y un "motor para el eje Y", sino que los movimientos son híbridos de movimientos entre ambos motores; Un movimiento en el eje X

se logra moviendo ambos motores en la misma dirección, mientras que un movimiento vertical se consigue moviéndolos en direcciones opuestas. Esta técnica no obstante está integrada directamente en el firmware de la placa Sanguinololu, que utiliza exactamente las fórmulas mostradas en la Figura 3.2.
En cuanto a la adaptación a una pecera, ha sido necesario diseñar y generar una colección de piezas específicamente diseñadas para ser montadas sobre los cristales de un acuario. El diseño de este conjunto de piezas fue el comienzo de este proyecto y sufrió varios cambios de rumbo en varios aspectos, tanto referidos a la estructura del sistema como a las plataformas de diseño.

A continuación se describen las herramientas de diseño 3D que se han utilizado, así como sus problemas y resultados.

3.2.1 Blender

En un principio se comenzó utilizando el software de diseño en 3D "Blender". Este entorno no obstante fue descartado al cabo del poco tiempo debido a que era inadecuado para la tarea que se le estaba dando: Blender pertenece a una fundación de software libre enfocada fundamentalmente a diseños artísticos y no mecánicos.
Existen varias alternativas centradas en el diseño industrial, como por ejemplo Solid Works pero continuando con la filosofía del software libre, se ha optado por utilizar OpenSCAD

3.2.2 OpenSCAD

OpenSCAD[27] es un software para la creación de modelos 3D. No obstante, OpenSCAD no es un programa de modelado interactivo, sino que es más parecido a un "compilador 3D" que lee un fichero de código que describe al objeto y renderiza a partir de él el modelo 3D (ver Figura 3.3).

Esto permite un control completo sobre el proceso de modelado y el cambio de cualquiera de los detalles, así como la creación de modelos paramétricos configurables en base a variables comunes. Esta ventaja ha sido clave en el diseño de AquaRide; todas las piezas se pueden modificar en minutos para adaptarse a nuevos acuarios o experimentos.

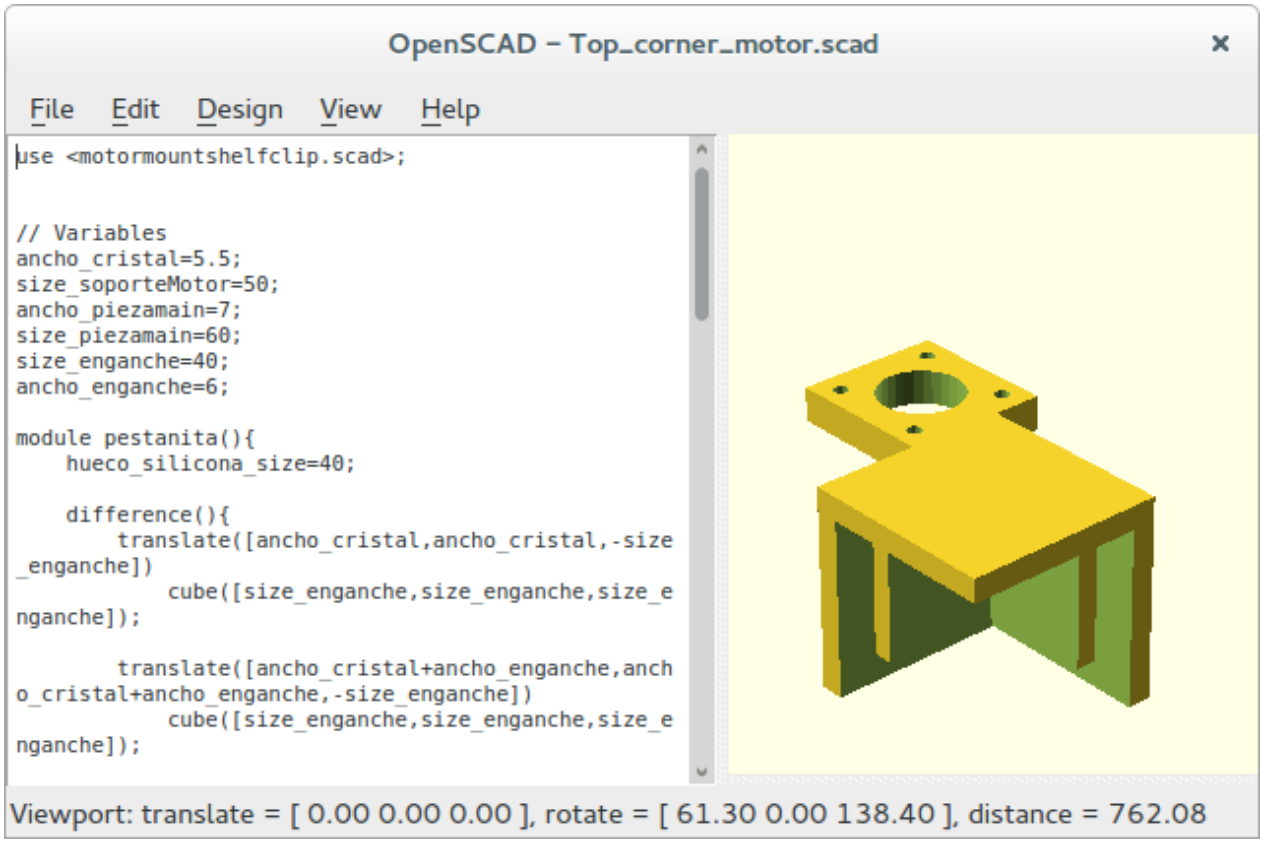


Figura 3.3: Interfaz de OpenSCAD. A la izquierda el editor de código y a la derecha el objeto renderizado a partir de éste.

3.2.3 Diseño final

El código de todas estas piezas diseñadas está recogido en el anexo F: Código utilizado: OpenSCAD.

Tras varias iteraciones en el proceso de diseño, y varios cambios en la estructura que se pensaba realizar, el sistema final está compuesto de:

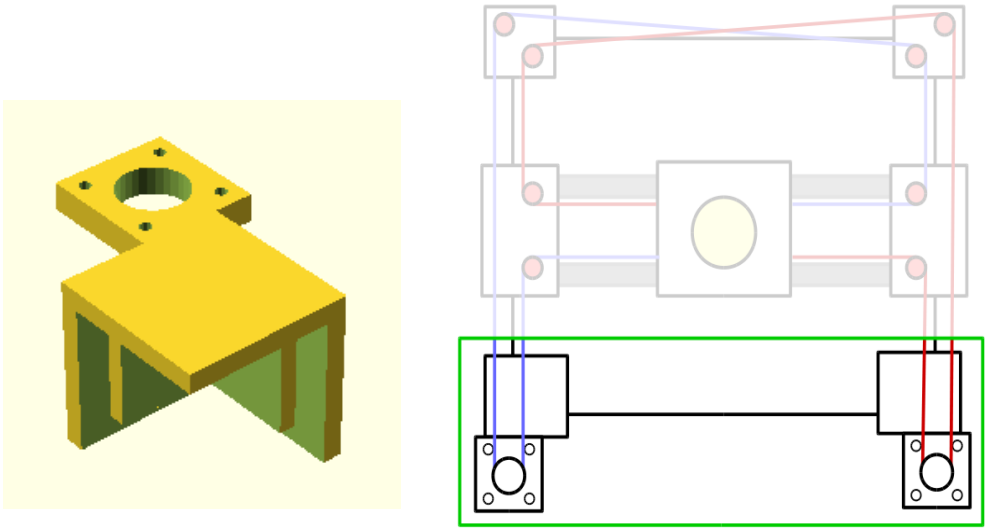


Figura 3.4: Pieza "Top_Corner_Motor.scad" y su posición equivalente en el esquema de CoreXY

- 2 Piezas para la sujeción de los motores. Estas piezas, situadas en las esquinas de uno de los laterales de la pecera, se encargan de sujetar con firmeza los motores y proporcionar una sujeción adecuada al acuario. Además, se utilizan para recoger el organizador de cables que recoge los mismos desde la parte móvil, evitando así que se sumerjan los cables en el agua. Se puede ver en la Figura 3.4.
- Dos ruedas adaptadas al grosor del eje de los motores paso a paso para transmitir la fuerza de éstos a los cables de Nylon. Se puede ver esta pieza en la Figura 3.5.

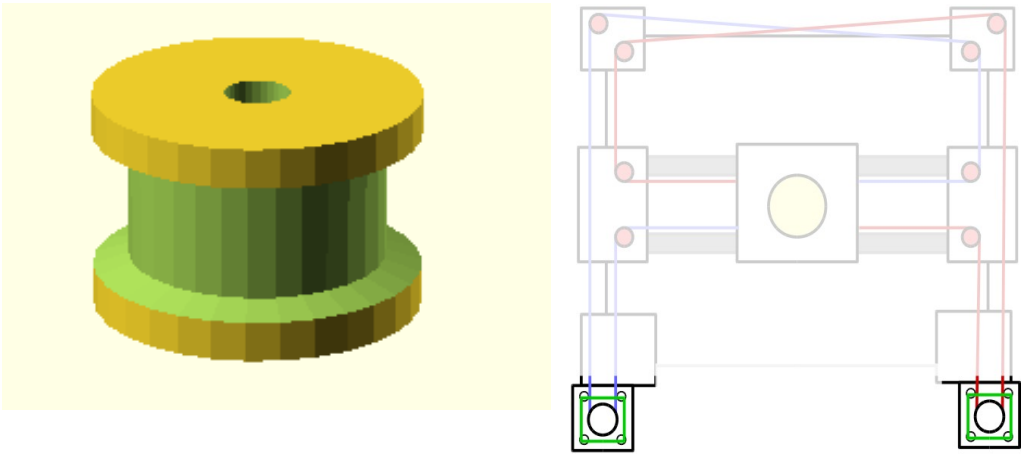


Figura 3.5: Pieza "Motor_Wheel" y su posición equivalente en el esquema CoreXY.

- 2 Piezas para el guiado del cable en las esquinas posteriores. En frente de las piezas anteriores, se sitúan estas piezas encargadas simplemente de actuar como guías para ambos cables. Se puede ver en la Figura 3.6 su forma y lugar de colocación.

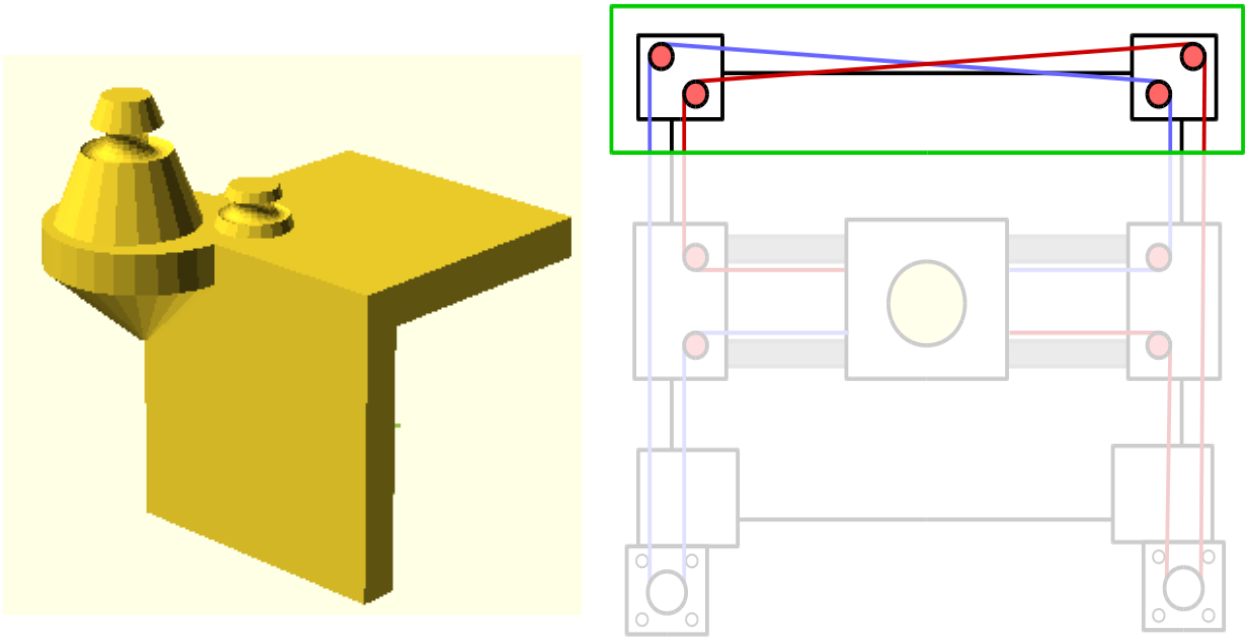


Figura 3.6: Pieza "Top_corner" y su posición equivalente en el esquema CoreXY.

- 2 Piezas laterales móviles que soportan el peso de la estructura central y guían los cables de Nylon. Se puede ver su forma y posicionamiento en la Figura 3.7.

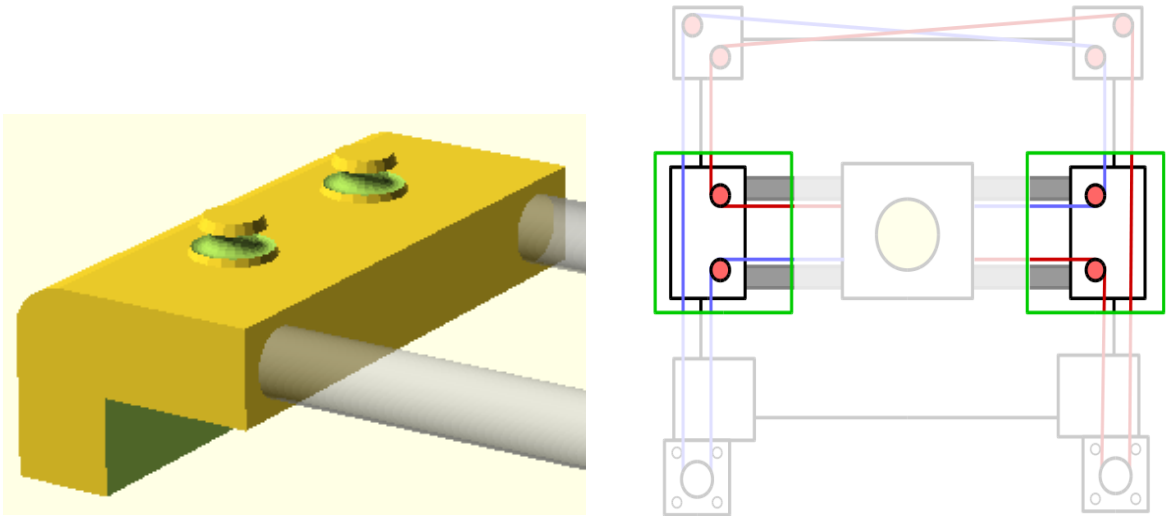


Figura 3.7: Pieza "Wagon" y su posición equivalente en el esquema CoreXY.

- 2 Barras de acero que soportan toda la estructura central, permiten su deslizamiento en el eje X y transmiten el movimiento en el eje Y. Se puede ver su posición en la Figura 3.8.

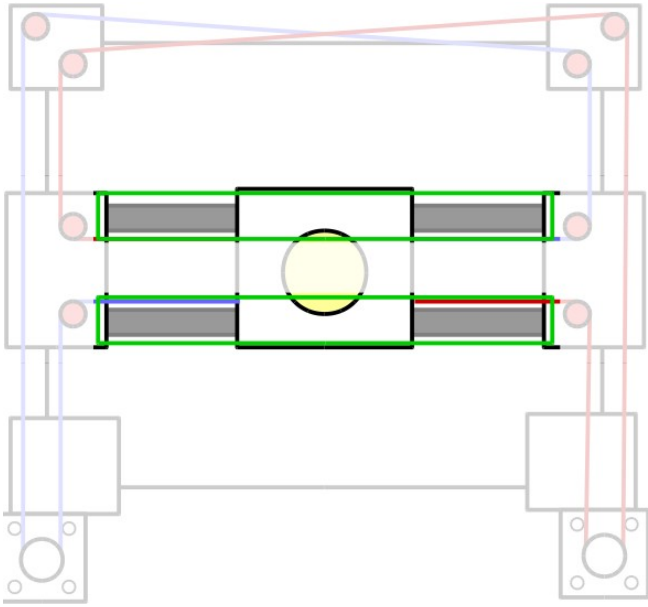


Figura 3.8: Posición equivalente de las barras de acero en el esquema CoreXY.

- Una plataforma móvil central en la que se encuentran el eje que sostiene a la sonda electro-receptora, el servomotor que gira el eje y el motor paso a paso que lo sube y baja. Esta pieza además es la encargada de transmitir las tensiones de los cables al sistema, y es a ella a la que se atan. Se puede ver en la Figura 3.9.

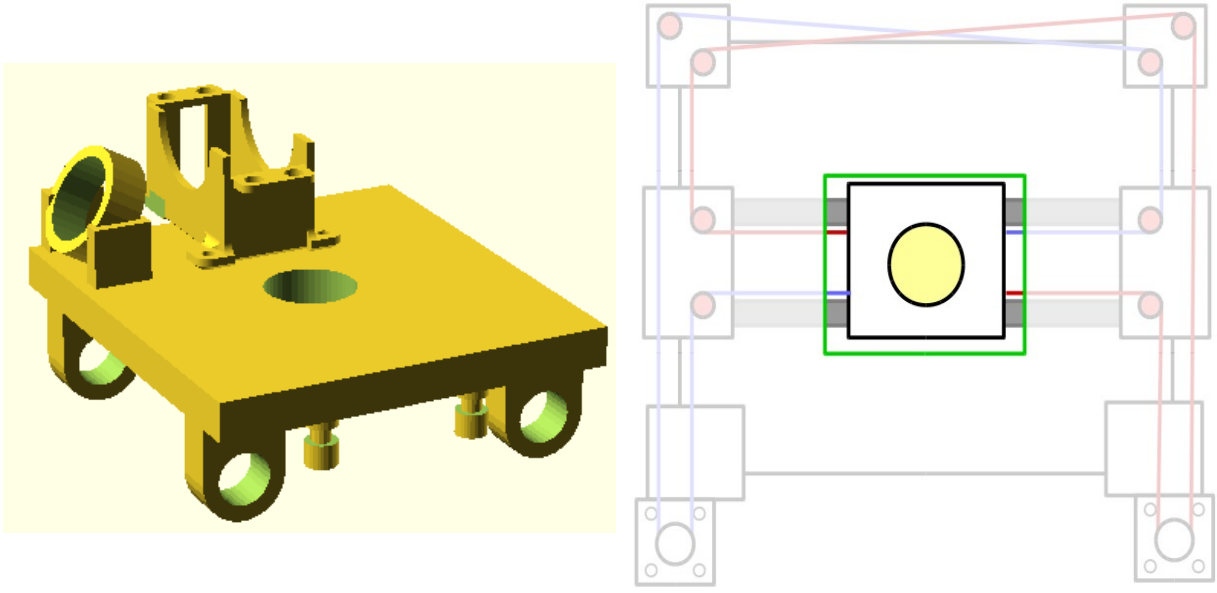


Figura 3.9: Pieza "Carrier" y su posición equivalente en el esquema CoreXY.

- Una pieza central, móvil y girable que permite el movimiento de la sonda en Z y en A. Además, soporta el motor paso a paso encargado de mover el eje dentado y el final de carrera del eje Z. Se puede ver en la Figura 3.10.

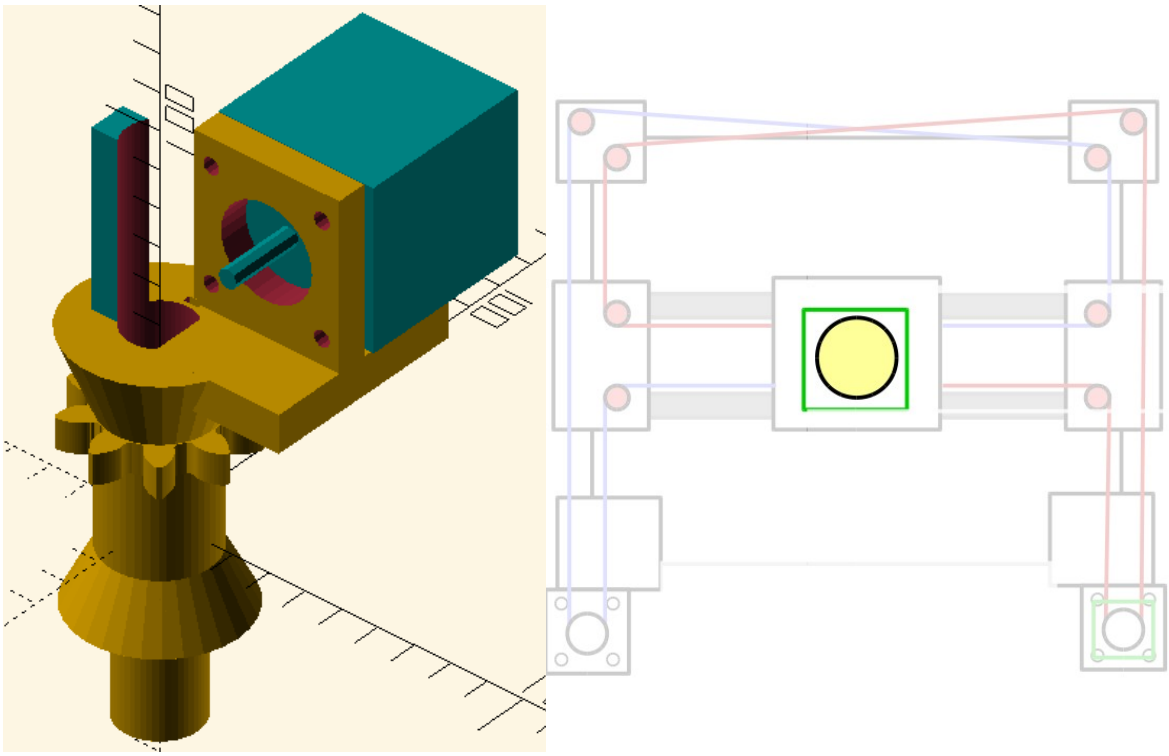


Figura 3.10: Pieza "Eje_transmision" y su posición equivalente en el esquema CoreXY.

- Un eje dentado hueco, que deslizará por dentro del eje de transmisión y que tendrá por dentro los cables pertenecientes a los electrodos de la sonda. Se puede ver en la Figura 3.11.

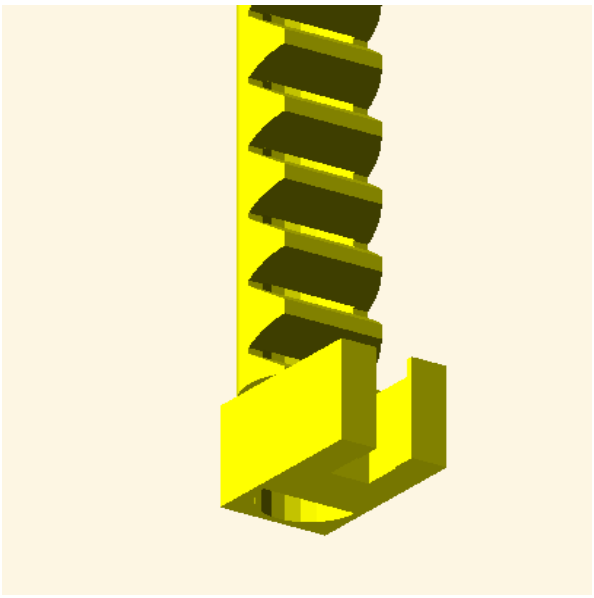


Figura 3.11: Pieza "Eje_Dentado" y su posición equivalente en el esquema CoreXY.

- Dos ruedas dentadas encargadas de transmitir la fuerza del servomotor y el motor paso a paso a las correspondientes piezas. Se puede ver en la Figura 3.12.

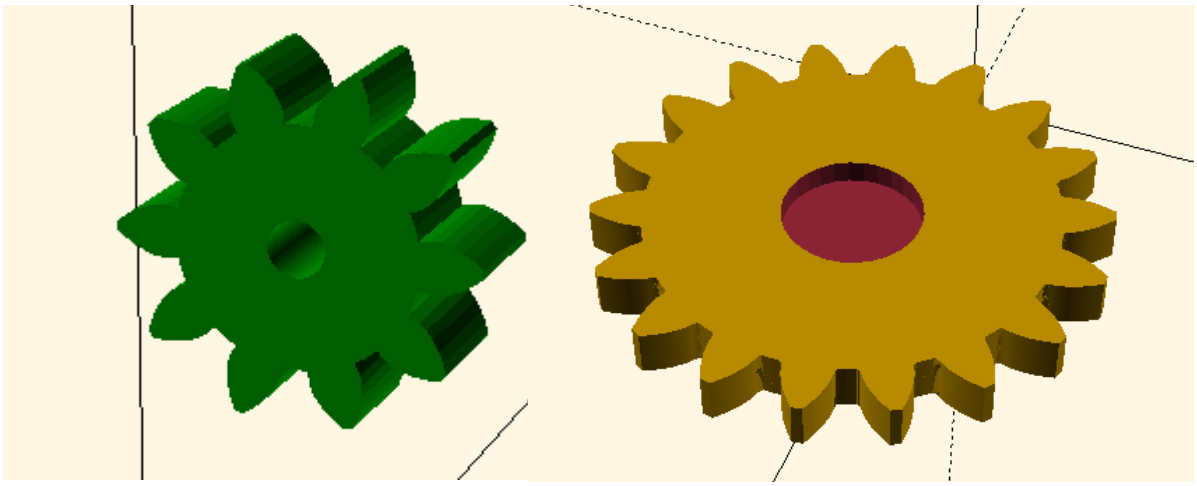


Figura 3.12: Piezas "RuedaStepperZ" y "RuedaServoA"

- Un conjunto de piezas que al combinarse con las arandelas, tornillos y tuercas de acero inoxidable correspondientes se convierte en la sonda electro-receptora, el núcleo de AquaRide. Se puede ver en la Figura 3.13.

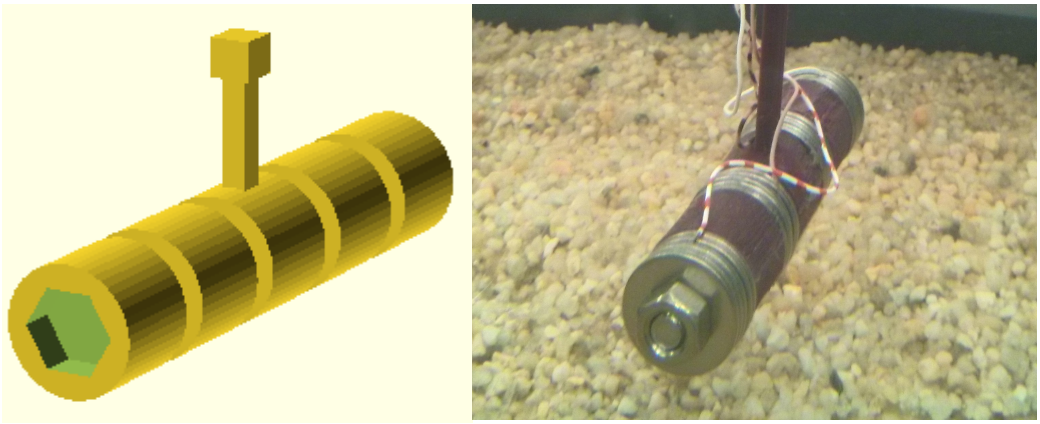


Figura 3.13: El sensor eléctrico. A la izquierda, la versión para imprimir y a la derecha, la pieza ya montada

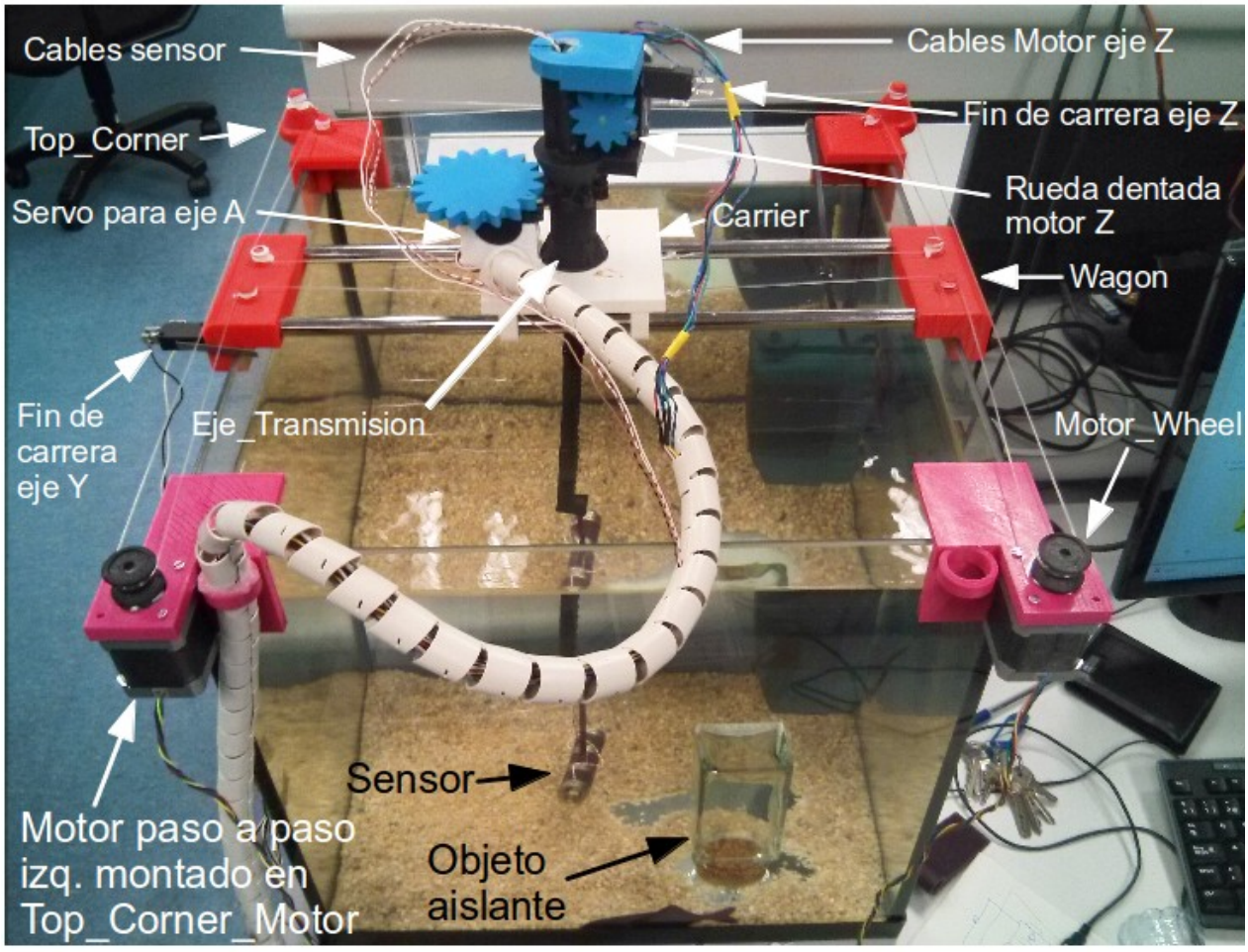


Figura 3.14: Montaje final de AquaRide. Vista superior frontal. Las piezas impresas en 3D están siguen la nomenclatura empleada en OpenSCAD y se pueden ver una por una en esta sección. En cuanto al sensor, se puede ver sumergido en el acuario, al lado de el objeto aislante utilizado en los experimentos. Este objeto es un bote de vidrio con forma de prisma cuadrangular regular, que se ha llenado con agua de la misma pecera para poder mantenerlo en el fondo durante los experimentos. Sus dimensiones son 6.5x6.5x12.5cm

Por último, se muestra una serie de imágenes (Figuras 3.14 y 3.15) del resultado final del montaje de AquaRide, indicándose la posición de las piezas descritas. Este montaje se ha realizado en una pecera normal de acuario con unas dimensiones de 44.3x44.3x45cm y un grosor de cristal de 0.5cm.

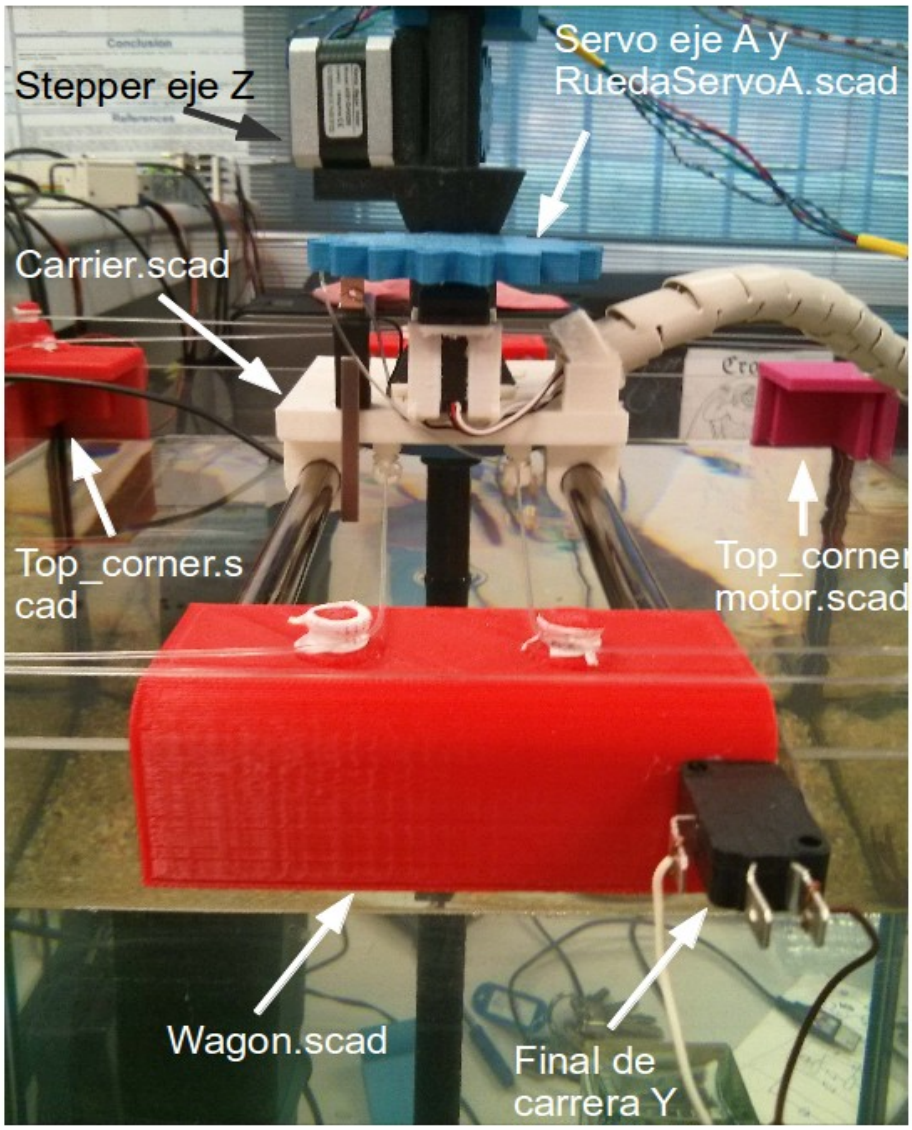


Figura 3.15: Vista lateral de la pieza impresa "Wagon.scad". En la parte de la derecha se puede ver el detector de fin de carrera del eje Y. Al fondo, en blanco, la pieza "Carrier.scad" impresa montada sobre las barras de acero. En esta pieza se puden atisbar el detector de final de carrera del eje X, el servomotor que mueve el eje A, su rueda dentada y la pieza "Eje_Transmision.scad" en negro, con el motor paso a paso que mueve el eje Z montado.

3.2.4 Errores en el diseño y correcciones

En este apartado se muestran los fallos e imprevistos del diseño y las soluciones propuestas.

1. Al diseñar la pieza dentada que sujeta directamente el sensor, no se tuvo en cuenta el límite para la altura que es capaz de fabricar la impresora 3D utilizada (BQ Prusa). A

consecuencia de esto se tuvo que dividir la pieza en tres secciones que fueron unidas mediante secciones especiales y pegamento, lo cual se traduce en que la pieza es más frágil de lo esperado.

- Esto se puede solucionar en el futuro o bien rehaciendo toda la estructura para permitir que esta pieza sea mas gruesa o encontrando una manera de imprimir en una sola pieza el modelo original con el tamaño total requerido.

2. Todo el sistema está pensado para utilizar hilos monofilares de Nylon., pero este tipo de cables sufre elongación térmica y sería recomendable utilizar, por ejemplo, un cable de acero.
3. Además, debido a que el cable de Nylon. produce quemadura por fricción (los soportes son estáticos), es necesario utilizar un cable de Nylon. de gran grosor para evitar que la quemadura se realice de manera más grave sobre un solo punto. Este efecto se observó al cabo de unas dos semanas de rodaje.
   - La solución más adecuada para este problema y el anterior es la sustitución de las conducciones de cable por poleas con rodamientos que al girar, eviten la fricción.
   - No obstante, la solución alternativa empleada es el cambio del cable de Nylon de grosor 0.5mm por uno de 1.1mm y el refuerzo de las conducciones con cinta de teflón para reducir la fricción.
4. En una primera instancia, todo el proyecto se diseñó para utilizar un servo encargado del movimiento en Z. Al implementar esta solución se observó que al necesitar un juego de ruedas dentadas que amplificaran el movimiento lineal realizado, estas reducían en gran manera el par del sistema, que era incapaz de levantar el sensor.
   - Para solucionar esto, se cambió radicalmente el diseño y se sustituyeron el juego de ruedas y el servomotor por un motor paso a paso y un detector de fin de carrera para el control de posición
5. Actualmente el sistema utiliza un servomotor para el control de la posición de giro. Debido a diversos factores, la pieza "Eje_Transmision" no mantiene exactamente su verticalidad sino que tiene una inclinación medida en el eje θ (ver Figura Figura 3.16) de aproximadamente 1-2º. Esto causa un cambio en el eje de giro de la pieza y dificulta la transmisión de fuerzas en el eje A.

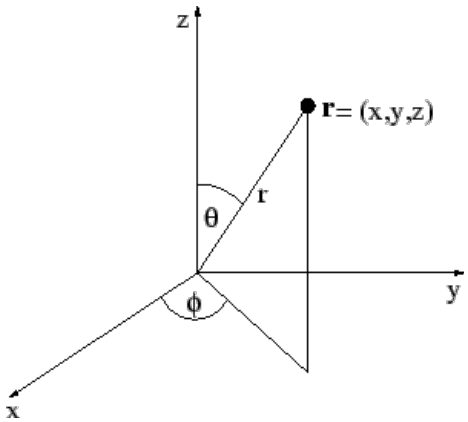


Figura 3.16: Coordenadas polares.

- Las dos soluciones propuestas para este problema son o bien cambiar el servomotor por un motor paso a paso que tenga un par mayor y mejor precisión en la posición ante inercias altas o bien mejorar la sujeción de la pieza que proporciona verticalidad a "Eje_Transmisión".

6. En la versión inicial, carrier.scad se desplazaba por ruedas encima de las varillas, se descartó por que la pieza que sostiene no tiene el centro de gravedad constante y podía volcar todo.
   - Este método se sustituyó por un deslizamiento a través de anillos que resultó ser muy efectivo.
7. Al cambiar el servomotor que se encargaba del movimiento en vertical por un motor paso a paso, se incrementó el peso de la estructura y por tanto la fricción con la pieza sobre la que gira ("Carrier.scad").
   - Este problema se ha solucionado añadiendo grasa lubricante en el eje de giro. Esto permite un giro suave y ligero.
8. La pieza "Eje_Dentado" no es suficientemente alta y el sensor no puede llegar hasta el fondo de la pecera.
   - Se ha reimprimido la pieza del sensor para hacerla más alta y de esta manera poder mapear si es necesario en el fondo del acuario.
9. Se utilizó pegamento termo-fusible para hacer todas las uniones de piezas y se descubrió al cabo de unos días que este tipo de uniones se degradan con facilidad.
   - Se ha sustituido el pegamento termo-fusible por pegamento de cianocrilato.
10. El receptáculo para el organizador de cables en la pieza "Carrier" se situó en un principio en una zona en la que no permitía correctamente el giro del sensor.
11. Debido a la fragilidad de la pieza dentada que soporta el sensor, se ha mantenido la velocidad de movimiento baja. Cuando esto sea corregido, se podrá aumentar con seguridad la velocidad de movimiento de los motores.

3.3 Hardware: La electrónica utilizada.

A continuación se hace un repaso por todos los componentes electrónicos utilizados.

3.3.1 Sanguinololu

Como nivel intermedio entre el sistema de desplazamiento mecánico CoreXY y el controlador implementado en Python, se ha utilizado un dispositivo electrónico en el que se han implementado las dos funcionalidades críticas del sistema, que son:

1. La gestión tanto de los servomotores como de los motores paso a paso y la implementación de los algoritmos de movimiento según la técnica CoreXY.
2. La adquisición de datos.

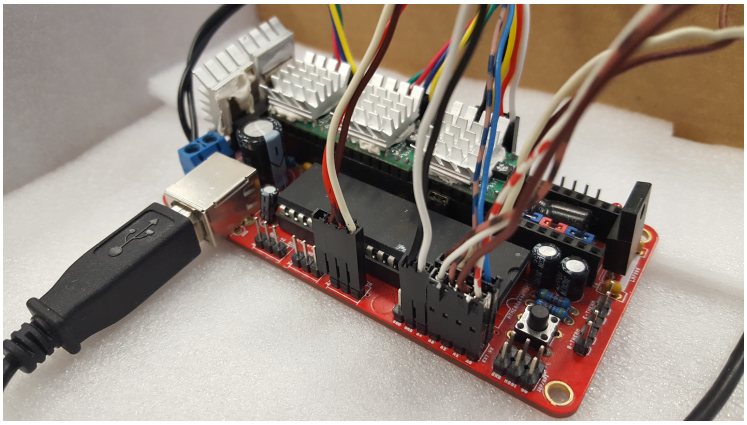


Figura 3.17: Fotografía de la placa Sanguinololu con las conexiones preparadas para el funcionamiento.

Sanguinololu es una solución electrónica todo en uno, de bajo costo para RepRap y otros dispositivos CNC. Presenta un clon de Sanguino a bordo utilizando el ATMEGA644P aunque un ATMEGA1284 puede ser fácilmente usado. Sus cuatro ejes son accionados por un controlador de pasos con pines compatibles con un Pololu. En este proyecto se han utilizado drivers Pololu A4988.

Las características de este sistema son las siguientes:

- Esta placa incluye un clon de Sanguino (una variante de la familia Arduino) y utiliza el ATmega644P de Atmel.
- Soporta el uso de hasta 4 Pololu stepper driver boards.
- Soporta múltiple configuraciones de alimentación.

  - Lógica & Motores alimentados por una fuente de poder ATX (necesita un conector Molex de disco duro, y un conector ATX 4P opcional, para una fuente adicional de 12 V).
  - La lógica es alimentada por el conector USB.

- Soporta múltiples configuraciones de comunicaciones
  - Conectividad USB por medio del FT232RL.
  - Conector USB2TTL disponible para cable FTDI, o modulo bluetooth BlueSMIRF.
- Dispone de 13 pines extra disponibles para expansión y desarrollo - 6 analógicas y 8 digitales, con las siguientes capacidades:
  - UART1 (RX y TX)
  - I2C (SDA y SCL)
  - SPI (MOSI, MISO, SCK)
  - PWM pin (1)
  - (5) Entradas / Salidas

3.3.2 Motores

Los motores utilizados son de tres tipos distintos:

1. 1x - Servomotor de 180º de rotación modelo Futaba S3003, utilizado en el giro del eje A.
2. 2x - Motor paso a paso modelo Wantai NEMA 17 de 1.8 grados/paso con un par de 4.8kg/cm, que corresponden a los motores izquierdo y derecho.
3. 1x - Motor paso a paso modelo Wantai NEMA 17 de 1.8 grados/paso y un par de 2.8 kg/cm. Este motor es el modelo Ultra Small que reduce su tamaño a costa del par y se utiliza en el eje Z, para evitar desestabilizaciones en el eje Z.

3.3.3 Detectores de fin de carrera

Como última pieza adicional del sistema, se han utilizado tres detectores de fin de carrera como medio de calibración de la posición del sistema de desplazamiento. En general, se puede utilizar cualquier tipo de detector de fin de carrera, pero en este proyecto se han utilizado los disponibles en el laboratorio, que eran compatibles con el diseño: los micro switches RLEIL de la serie RL6. Se especifica el modo de funcionamiento de estos detectores en el anexo B.

3.4 Hardware: El sensor electro-localizador

De cara a realizar el proceso de detección de objetos, es necesario conocer de manera teórica cómo se debe comportar el sistema para ser capaces de analizar los resultados de manera correcta. A continuación se muestra el proceso seguido para la construcción de un sensor que permitiese una lectura adecuada de las variaciones de campo frente a objetos.

3.4.1 Esquema tipo V-V

Tal y como se explicó en la introducción, frente al esquema tipo V-I de medida de corrientes, se ha decidido utilizar un modelo tipo V-V (ver Figura 3.18), más similar al utilizado por A. MacIver [17].

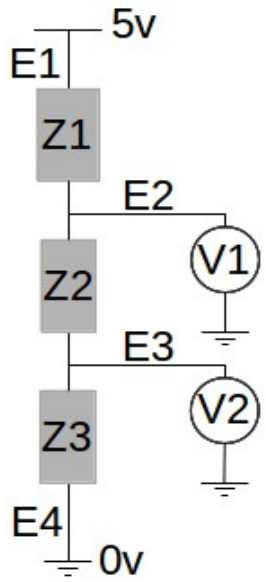


Figura 3.18: Circuito eléctrico equivalente al modelo de medidas tipo V-V. Se han indicado los nombres de los nodos asociados a los electrodos E1-E4. Para realizar una comparativa con los modelos reales utilizados, ver Figura 3.19.

En este modo de operación se utiliza un electrodo de emisión, uno de tierra y un electrodo por cada punto que se desee medir, en el que se sitúa un voltímetro de alta impedancia. De esta manera, se evita un conflicto entre medidas y se pueden realizar varias de manera simultánea, permitiéndose de esta manera aumentar la complejidad de los sensores sin tener que cambiar el modelo de medida, como se explica en la sección 3.4.3 "El algoritmo de medida".

Como se puede ver en la Figura 3.19, en el sensor empleado para AquaRide se han utilizado un total de cuatro electrodos, de los cuales dos se han empleado en la emisión de pulsos y dos se han empleado en la medida de voltaje.

Es importante destacar que Z1, Z2 y Z3 son impedancias que representan la suma de los efectos del agua y los posibles objetos. Se han representado como impedancias debido a que las frecuencias a las que se trabaja con las señales no pertenecen a la banda de frecuencias en las que el agua se comporta de manera puramente resistiva [16]. Esto representa un problema a la hora de evaluar el comportamiento del campo eléctrico, y más adelante se verá cómo esto afecta a las medidas realizadas y cómo se ha evitado su efecto en este sistema.

También es necesario aclarar que las representaciones que se hacen de datos a continuación dan como valores de voltaje los obtenidos por la función nativa de Arduino "AnalogRead", que hace la conversión del rango de voltajes admitido [0,5v] a los valores digitales [0,1024]. Debido a la naturaleza del sensor, no es necesario realizar ningún tipo de conversión, dado que el sistema se centra en analizar los cambios en el medio y no es necesario para esto usar los valores analógicos.

Por lo tanto, en las gráficas a continuación, un valor de 1024 representa 5 voltios y por ejemplo uno de 512 representaría 2.5 voltios medidos en el electrodo en cuestión.

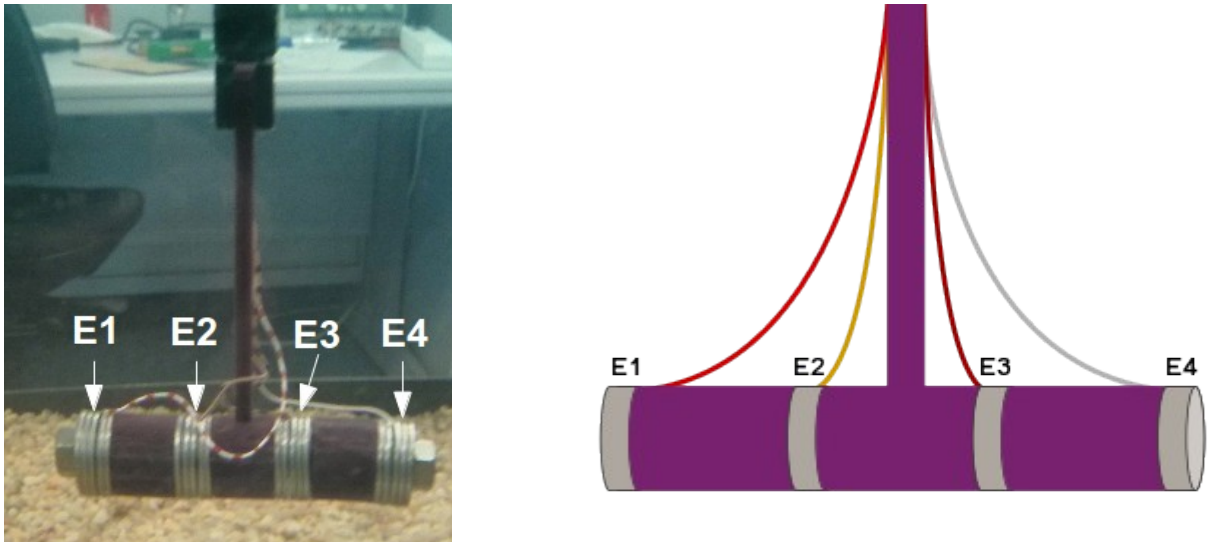


Figura 3.19: Sensor construido a la izquierda y esquema original de diseño a la derecha. En cuanto a la estructura interna del sensor, las partes representadas en morado son piezas de plástico PLA aislantes, y los electrodos no están conectados internamente.

El algoritmo empleado para realizar las medidas está programado en la placa Sanguinololu y los electrodos están conectados directamente a sus pines analógicos (para ver detalladamente las conexiones de estos pines, acudir al anexo B: Conexiones y montaje). Esto es posible gracias a que la resistividad del agua es considerable (centenas de KΩ en este caso, pero depende exponencialmente de la distancia entre electrodos) y las corrientes que se generan son bastante menores de lo máximo permitido por los pines de la placa Sanguinololu (40mA). Esto podría representar un problema en el caso de que un par de electrodos de emisión-recepción estuvieran demasiado cerca, ya que el pin de la placa no soportaría dar tanta corriente al reducirse la resistencia entre fuente y tierra demasiado.

Otro problema contrario al presentado es que los electrodos tampoco pueden estar excesivamente lejanos, ya que esto aumenta mucho la resistencia existente entre ambos, reduciendo por tanto la corriente, y el problema se presenta al realizar las medidas, ya que el conversor analógico-digital drena corriente al hacer muestreos, y habría problemas si no hay suficiente corriente. No obstante, este problema no se ha observado en el sistema construido.

A continuación se muestra el algoritmo más simple de muestreo utilizado:

1	`Estado de reposo`	`E1=E4=0V`	
2	`Emisión a través de E1`	`E1=5V y E4=0V`	Lectura de voltajes en E2 y E3, que dan lugar a las medidas V10 y V20
3	`Estado de reposo`	`E1=E4=0V`	

Tabla 1: Bucle simple inicial para la toma de muestras en una posición estática.

Como se puede entender de este algoritmo, el tipo de señal que se está emitiendo es un pulso de tipo cuadrado, en el que se realizan muestreos a lo largo de su duración. Estas señales, en el dominio de la frecuencia tienen componentes en todo el espectro. Por este motivo, es necesario tener en cuenta cómo afecta la impedancia del entorno al campo generado. Para ello, se ha realizado de manera experimental una emisión periódica de pulsos a través de los pines de salida

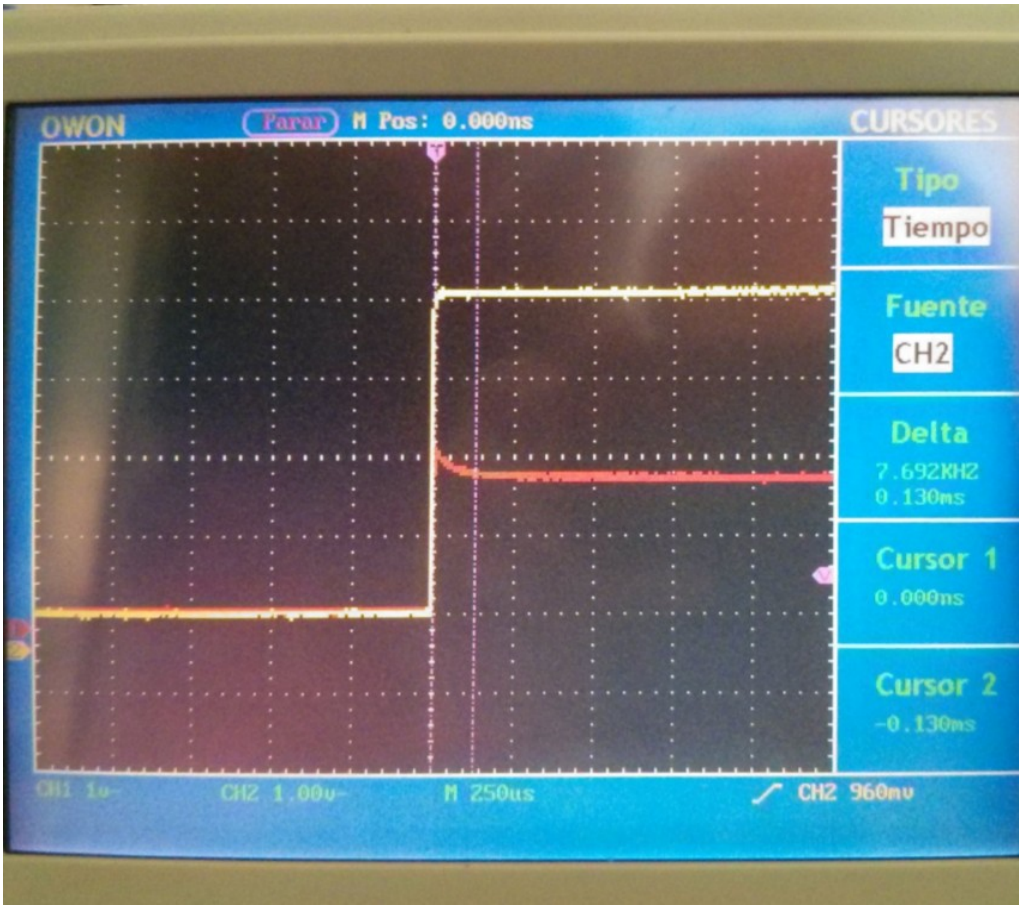


Figura 3.20: Detalle del comportamiento no resistivo del medio. En amarillo el canal 1 del osciloscopio que representa el pulso emitido visto en los pines de salida de la placa Sanguinololu. En rojo, la señal de voltaje entre uno de los electrodos intermedios de medida (E2) y tierra. En la gráfica se ha centrado la vista en el transitorio de subida del pulso. Como se puede ver, el transitorio correspondiente a las componentes capacitivas/inductivas termina aproximadamente a los 200µs, momento a partir del cual se trata al medio simplificándolo como resistencias en lugar de impedancias, utilizando los conceptos básicos de sistemas de control estudiados.

de la placa Sanguinololu (ver Figura 3.20). Gracias a este experimento, se pudo comprobar cual era el tiempo de establecimiento de la señal para este caso, siendo éste cercano a los 200µs. Para asegurar no obstante que este efecto no causara problemas, se han realizado todas las medidas 1ms después del cambio en el valor de alguno de los pines de emisión, durante la emisión por tanto de cada uno de los dos pulsos emitidos por ciclo de medidas (el algoritmo se explica más en detalle en la sección 3.4.3).

3.4.2 Ruido en las medidas

Hasta el momento, no se ha tenido en cuenta que debido a la naturaleza dinámica del agua, las interferencias producidas por el medio y los componentes eléctricos que forman el sistema, existe un ruido en las señales muestreadas por los conversores analógico-digital de la placa. Para caracterizar este ruido, se procedió a la realización de un experimento en el que se tomaron un número elevado de muestras correspondientes a un mismo medio estático. Este experimento dio como resultado las señales que se pueden ver en la Figura 3.21.

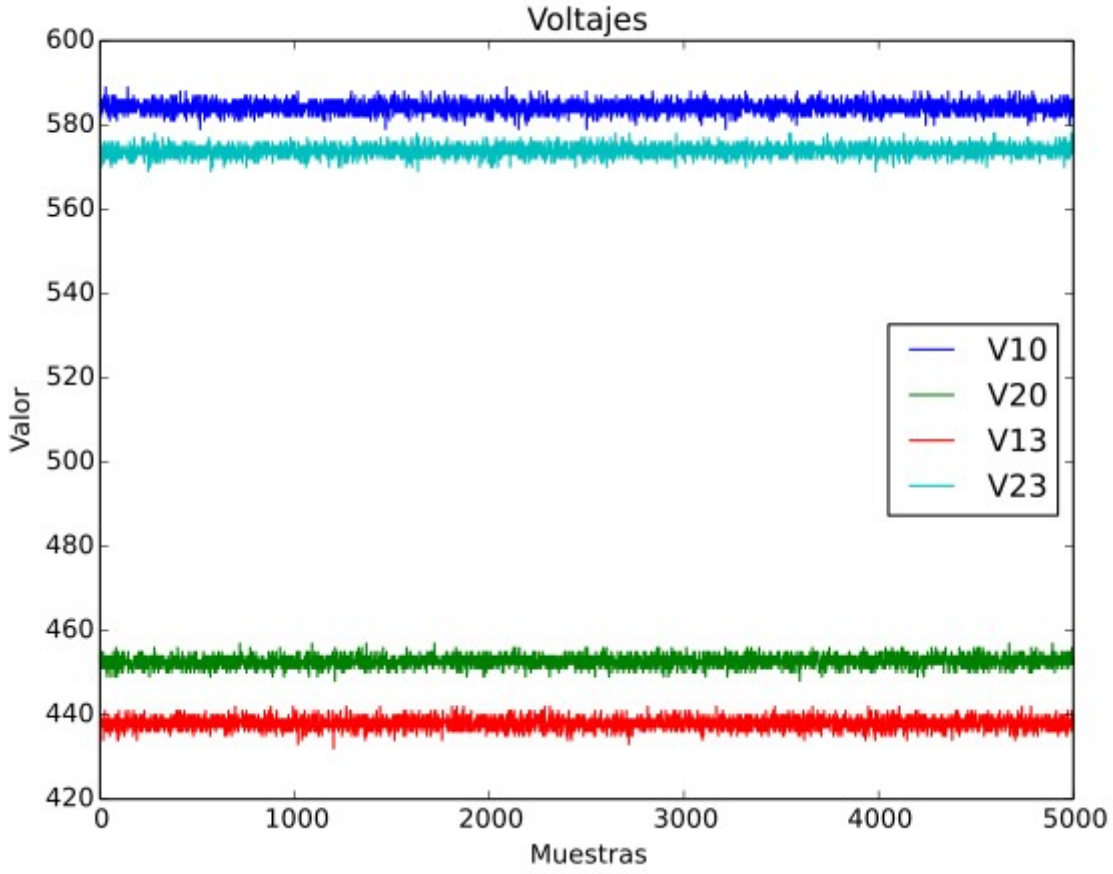


Figura 3.21: Muestras obtenidas en medio estático. Representación realizada con la librería Matplotlib de Python.
En el siguiente apartado se verá el significado de las distintas gráficas.

Para poder ser capaces de filtrar este ruido, fue necesario generar un histograma para cada uno de estos sensores. Como se puede ver en el histograma de la Figura 3.22, el ruido en las medidas tiene forma de distribución gaussiana estrecha, con una oscilación máxima pico a pico de 8 puntos. El eje representado como "Valor" indica la lectura realizada por el ADC de la placa, teniendo un rango de [0,1024] para el rango de voltajes recibidos de [0,5V].

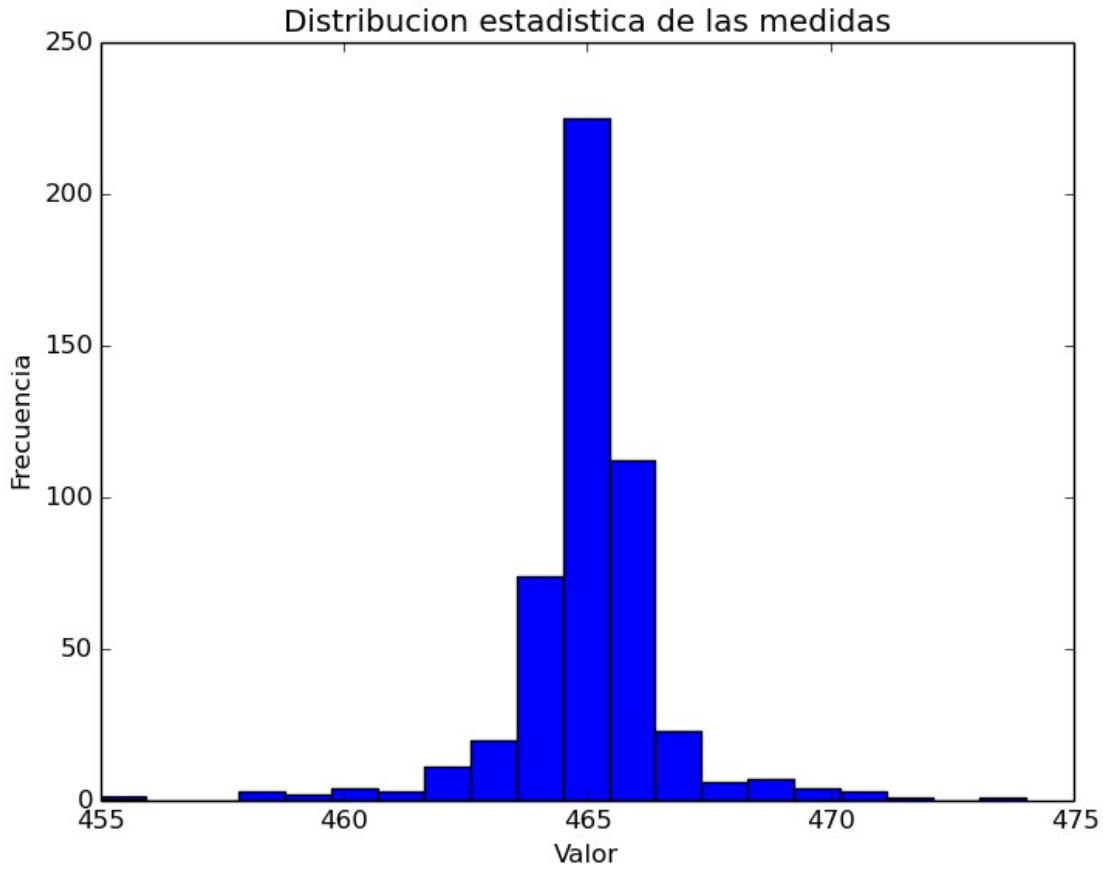


Figura 3.22: Histograma: Desviación de los valores obtenidos para un bloque de cinco mil muestras. Se muestra el histograma correspondiente a la medida V10, aunque es del mismo tipo para las cuatro.

Tras caracterizar el ruido del sistema, se decide que la manera más sencilla de filtrar este ruido es obteniendo una serie de muestras y calculando el valor medio de la serie. Para calcular el número de muestras necesario para calcular la media, se estima utilizando la Figura 3.23.

Como se puede ver a partir de las 100 muestras la media se estabiliza y el valor calculado es consistente en el tiempo, y en ningún caso se supera un valor de desviación con respecto al valor final de 1. Por este motivo, se puede deducir que para medias calculadas con valores mayores que 100, el valor obtenido como media va a ser muy similar al valor final hallado, con una precisión mayor que un salto del ADC.

Esto quiere decir que en este caso, el cuello de botella en cuanto a obtener unas medidas más precisas es la resolución del ADC, que introduce un error de cuantificación no despreciable.

Valores inferiores a este en el número de muestras empleado producirán, en el peor de los casos, saltos en el valor obtenido de 1, lo cual representa un único salto en la medida realizada por el ADC. Esto produciría granularidad en los resultados, en los que habría de manera aleatoria variaciones de +-1 salto en la medida con respecto al valor correcto. Debido a que el ADC de Sanguinololu trabaja con un intervalo de 1024 valores para el rango de 0 a 5 voltios, esta granularidad tendría un valor de salto de 0.005 voltios aproximadamente.

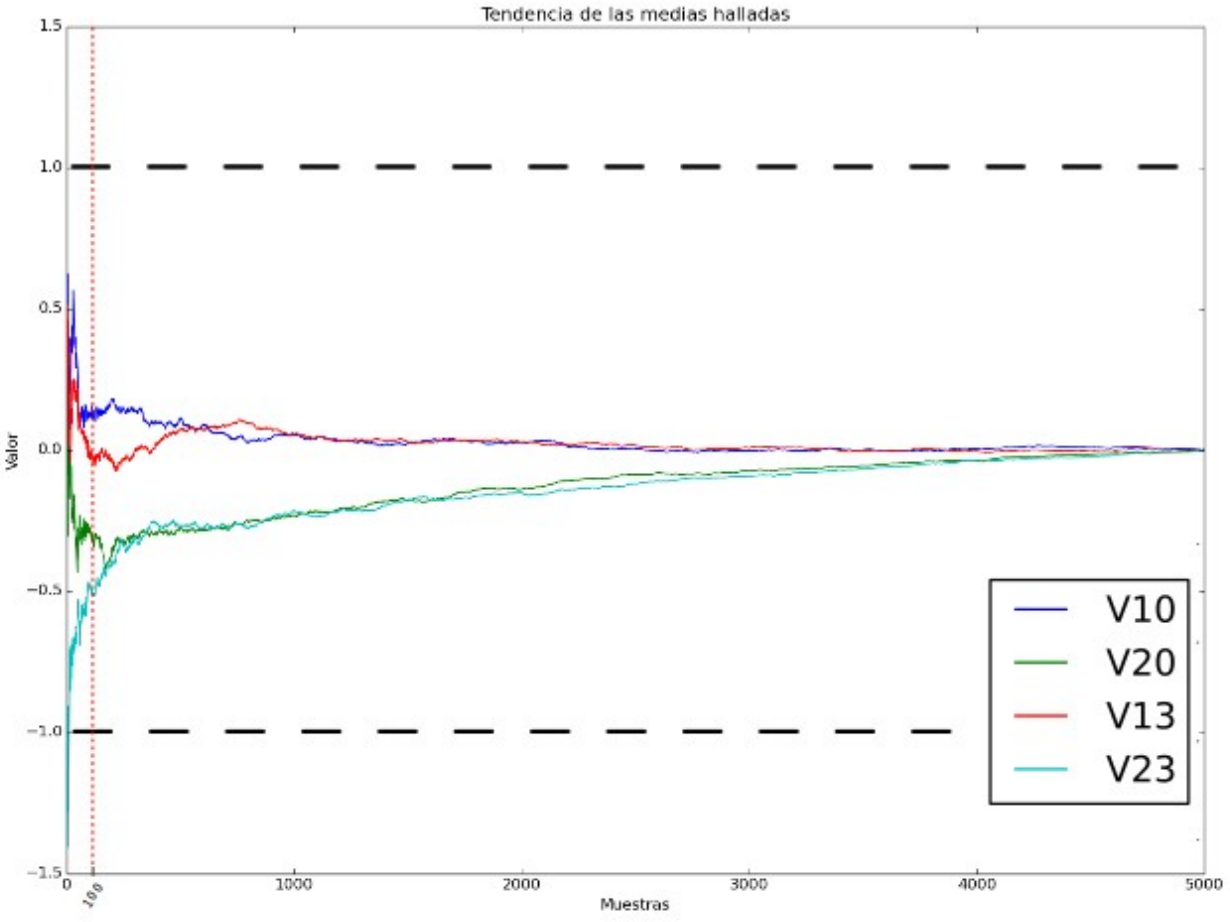


Figura 3.23: Sobre las 5000 muestras, resultado de calcular la media de los primeros N valores, restando el valor final a todos los resultados para poder realizar la comparativa.A partir de las primeras decenas de muestras el valor ya cae en el rango [+1,-1], y a partir de las 100 ya es suficientemente estable.

Este tipo de error, aunque solventado, no supone un problema grave ya que las variaciones producidas por objetos tienen una amplitud (dependiendo de la distancia) unas diez veces más grande. Por este motivo, en el caso en el que se plantee una mejora enfocada a realizar medidas en movi-

miento, este efecto no representa un problema ya que los bloques de medidas necesarios para calcular valores periódicamente se podrían realizar cada aproximadamente 100ms, lo cual dependiendo de la velocidad de tránsito puede ser perfectamente aceptable.

A partir de este punto del documento, todos los resultados que se muestren aplican ya el filtrado de ruido con un número de muestras utilizado de N=300, a no ser que se especifique lo contrario.

3.4.3 El algoritmo de medida

El algoritmo que se ha usado en los experimentos finales no es exactamente como el explicado al principio de este capítulo, sino que es ligeramente más complejo y se aprovecha de la capacidad de la placa Sanguinololu de cambiar los valores de sus pines de salida. Para aumentar la información que se puede obtener del medio alrededor del sensor, se ha decidido implementar por tanto el siguiente algoritmo:

#	**Acción realizada**	**Valor de los electrodos de emisión**	**Electrodos medidos**
1	Estado de reposo	E1=0V y E4=0V	
2	Emisión a través de E1	E1=5V y E4=0V	
3	Lectura de voltajes	E1=5V y E4=0V	V10(en E1) y V20(en E2) (N=300)
4	Estado de reposo	E1=0V y E4=0V	
5	Emisión a través de E4	E1=0V y E4=5V	
6	Lectura de voltajes	E1=0V y E4=5V	V13(en E1) y V23(en E2) (N=300)
7	Estado de reposo	E1=0V y E4=0V	

Tabla 2: Algoritmo de toma de muestras. N es el número de muestras tomadas, cuyo valor se ha concluido en el apartado anterior tras analizar el ruido del sistema.

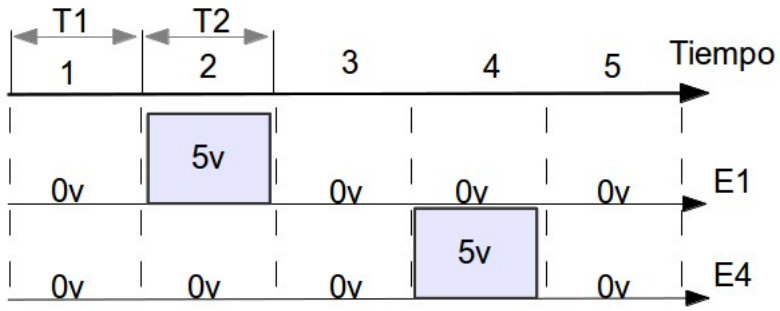


Figura 3.24: Diagrama de tiempos. Representación de un ciclo de medidas con los valores de E1 y E4. El valor de T1 es aproximadamente 1ms y el valor de T2 es de 1ms multiplicado por N, siendo N el número de muestras tomadas del experimento para cada posición.

Esta información obtenida a partir de la estimulación simétrica de ambos electrodos laterales (ver Figura 3.24 y 2) permite aumentar la precisión en la detección de objetos, y discriminar mejor factores como direcciones de aproximación. A continuación una tabla que muestra la nomenclatura utilizada en el código programado, y en las figuras 3.25 y 3.26 se puede ver cómo afectan las distintas emisiones al circuito simplificado.

Electrodo medido	**Electrodo a 5v**	**Electrodo a 0v**	**Nomenclatura**
E2	E1	E4	V10
E3	E1	E4	V20
E2	E4	E1	V13
E3	E4	E1	V23

Tabla 3: Nomenclatura utilizada para las distintas medidas de los electrodos

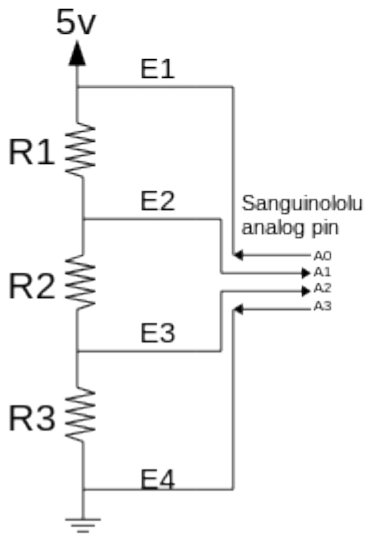


Figura 3.25: Circuito simplificado del sensor durante la fase 2

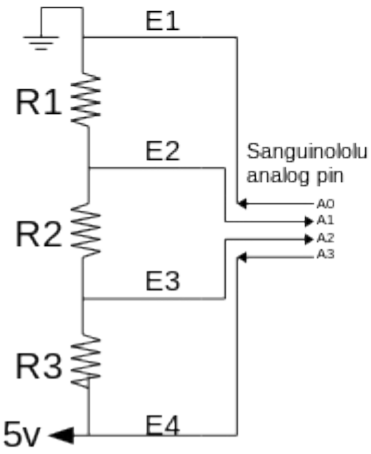


Figura 3.26: Circuito simplificado del sensor durante la fase 4

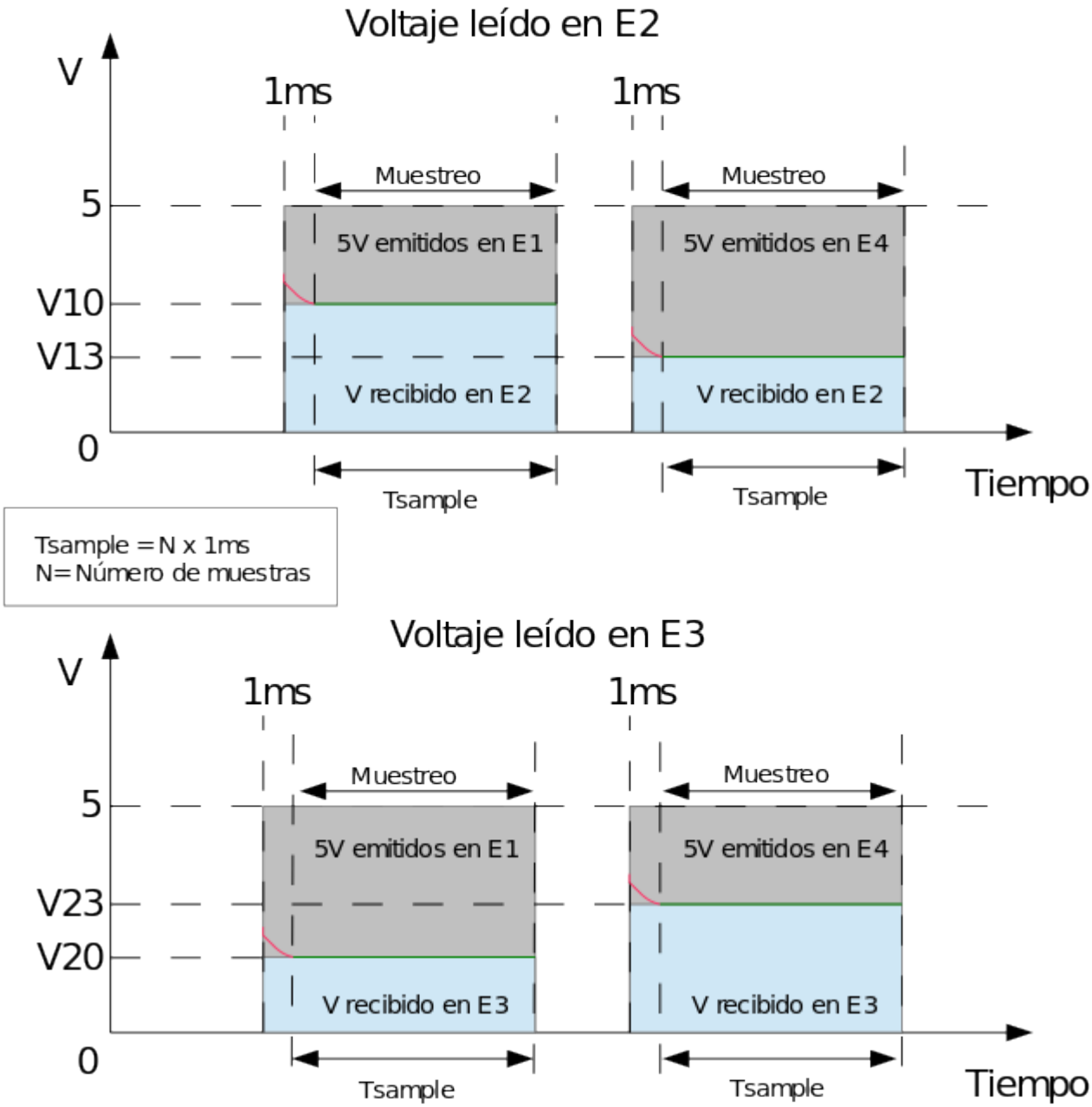


Figura 3.27: Diagrama de los pulsos emitidos y recibidos en el ADC. Se muestran tanto los tiempos de muestreo como los de espera. En gris se muestran los pulsos emitidos desde los electrodos de emisión E1 y E4. En azul, el pulso recibido en E2 y E3 para las representaciones superior e inferior respectivamente. La fase de muestreo tendrá un ancho variable directamente relacionado con el número de muestras que se requieran, teniendo en cuenta que entre muestra y muestra se deja 1ms de espera para estabilizar de nuevo el sistema, por precaución. En rojo se destaca el comportamiento no resistivo que se trata de evitar con el período de espera, como el visto en la Figura 3.20.

3.4.4 El modelo eléctrico

El modelo eléctrico que se aplica para entender este sistema es el de un dipolo simple con las cargas distanciadas una distancia "d" (ver Figura 3.28), que en este caso es fija para los dos tipos de estimulación, que lo único que hacen es cambiar la polaridad del dipolo.

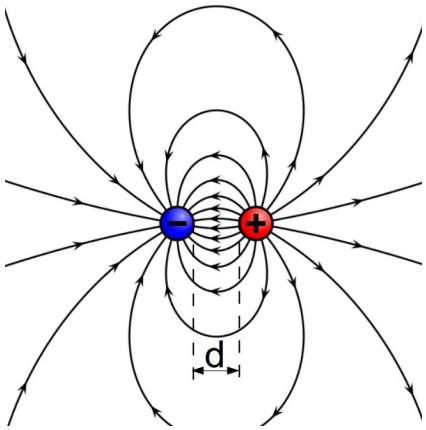


Figura 3.28: Dipolo eléctrico con separación entre polos "d"

Tal y como se vio en la sección 3.4.2, los muestreos se realizan 1ms después del cambio de cualquier valor en los pines de estimulación de la placa, tras pasar el tiempo de establecimiento. Por lo tanto, se puede representar el circuito en el momento de los muestreos como un divisor de tensión en el que las impedancias se transforman en simples resistencias (ver Figura 3.29).

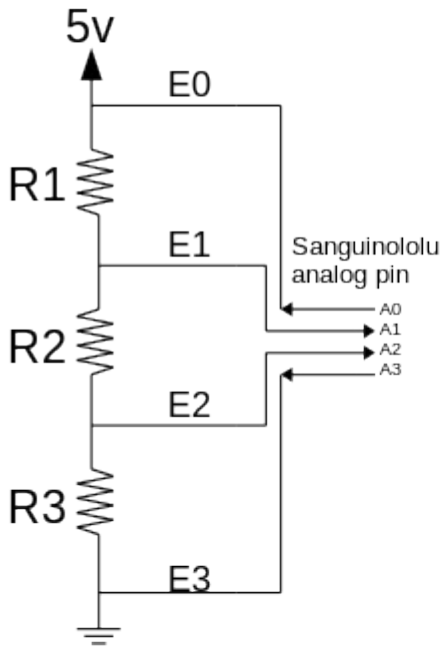


Figura 3.29: Esquema de estimulación, donde las impedancias han sido sustituidas por resistencias tras las conclusiones sacadas en el apartado 3.4.2.

Además, hay que tener en cuenta que al ser cada medida independiente de las demás, se puede dividir y analizar cada esquema de medida por separado, como se muestra en las figuras 3.30 y 3.31. En este caso, el diseño del sensor permite evaluar la diferencia entre estas dos lecturas, pero en casos futuros donde el sensor disponga de muchos más puntos de medida, este tipo de

Si se analiza con más detenimiento este esquema, se puede ver que es un divisor de corriente típico, en el que las resistencias R1 y R2+R3 varían según la resistividad del medio. R1 aumentará por tanto si hay un objeto aislante interfiriendo con las líneas de campo del dipolo y situado entre los electrodos E1 y E2, lo que provoca una caída en el voltaje recibido en E2. Por otro lado, ese mismo objeto aislante, situado entre E2 y E4 provocará un aumento del conjunto de resistencias R2+R3, lo que provocará un aumento en el potencial medido en E2.

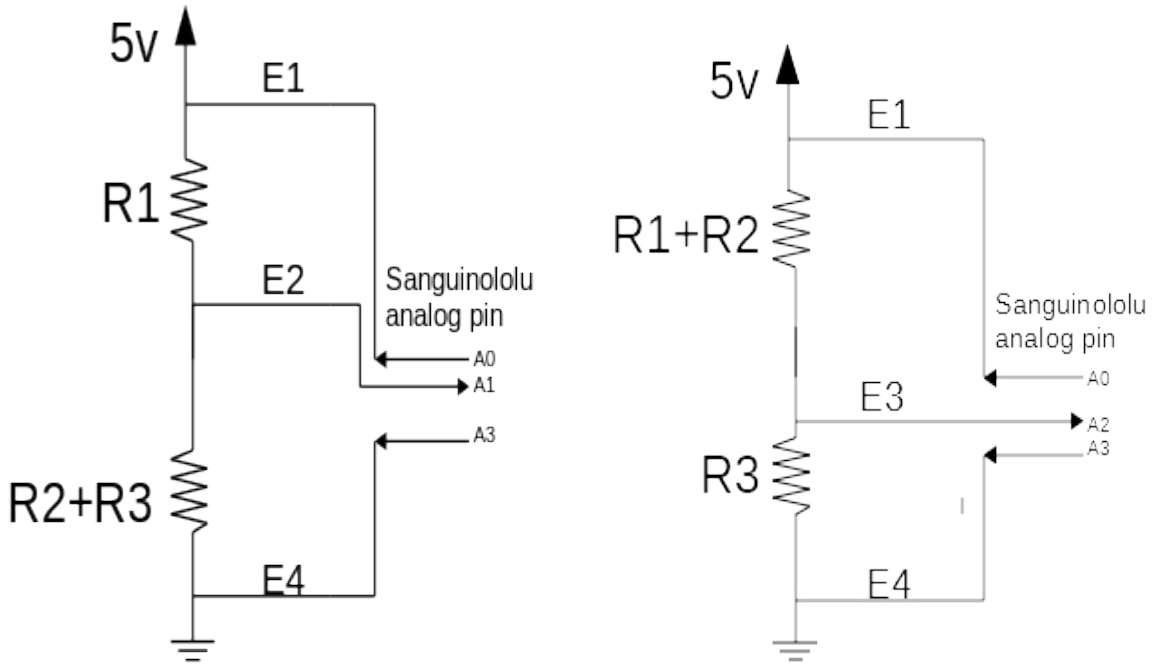


Figura 3.30: Ejemplo de lectura de electrodo E2 (izquierda) en el que las resistencias en serie R2 y R3 se han simplificado. Este esquema representaría el valor leído como V10 (ver 3). A la derecha, el caso contrario en el que se mide el electrodo E3, y las resistencias simplificadas son R1 y R2

Este efecto es de especial interés ya que no sólo permite detectar la presencia de un objeto en un área cercana al sensor, sino que, con cierto grado de incertidumbre, es posible localizar al objeto en una zona determinada. Con un número mayor de electrodos es posible aprovechar este efecto para realizar una representación del entorno.

3.5 Firmware: Programación de la placa Sanguinololu

A continuación se describe el proceso de diseño y la funcionalidad desarrollada para el software que dirige AquaRide [26]. El código de todo el firmware está disponible en el anexo E: Código utilizado: Firmware para la placa Sanguinololu.

3.5.1 Firmware libre

El primer instinto al abordar el problema de control de movimientos en este proyecto es la decisión en cuanto al firmware a utilizar. Debido a que la placa elegida está optimizada para el uso de impresoras 3D, la primera opción es el uso de firmware libre ya desarrollado y optimizado para impresoras 3D. En este aspecto, hay tres opciones compatibles: *Sprinter*, *Marlin* y *Teacup*.

Estas alternativas son muy efectivas pero tienen un importante problema de cara a su utilización en proyectos que difieren del uso específico de impresoras y es que al estar perfectamente optimizados, admiten muy poca variación en su configuración.

En este caso el diseño inicial incluía dos servomotores para los movimientos de giro y vertical, y acomodar el firmware para poder utilizar estos servos requería un análisis detallado de estos firmwares, ya que era necesario hacer varias modificaciones. Adicionalmente, en la placa Sanguinololu utilizada se debía incluir el filtrado de ruido y la adquisición de datos, lo cual requiere un espacio en memoria no despreciable y que fuerza al diseñador a aligerar en la medida de lo posible el código referido a gestión de la posición.

Por estos motivos, se decidió implementar un firmware nuevo específico para AquaRide, con las funciones necesarias para la gestión de los motores lo más simplificadas posible, y de esta manera tener un control absoluto de los procesos que se realizan. Esto además tiene una ventaja transversal, y es que en caso de necesitar cambios tanto en los algoritmos como en funciones mas elementales, la evolución del firmware es más rápida.

3.5.2 Firmware desarrollado para la placa Sanguinololu

El firmware programado en la placa (Anexo E:Código utilizado: Firmware para la placa Sanguinololu) se encarga de tres funciones principales:

- El movimiento del sensor, utilizando para ello motores paso a paso y servos.
- La toma de medidas. La placa se encarga de gestionar cómo se realiza la emisión de pulsos, el muestreo de los valores automáticamente y el envío de datos. Además, el firmware también realiza un filtrado de ruido previo al análisis.
- La interfaz. AquaRide puede ser fácilmente controlado manualmente por un usuario o por un programa gracias a una interfaz que permite el envío de comandos por puerto serie.

La ventaja principal de incluir estas tres tareas integradas en el mismo dispositivo es que existe una sincronización perfecta entre los valores de posición y los datos adquiridos en esos puntos, lo cual facilita mucho el procesado posterior de los datos obtenidos.

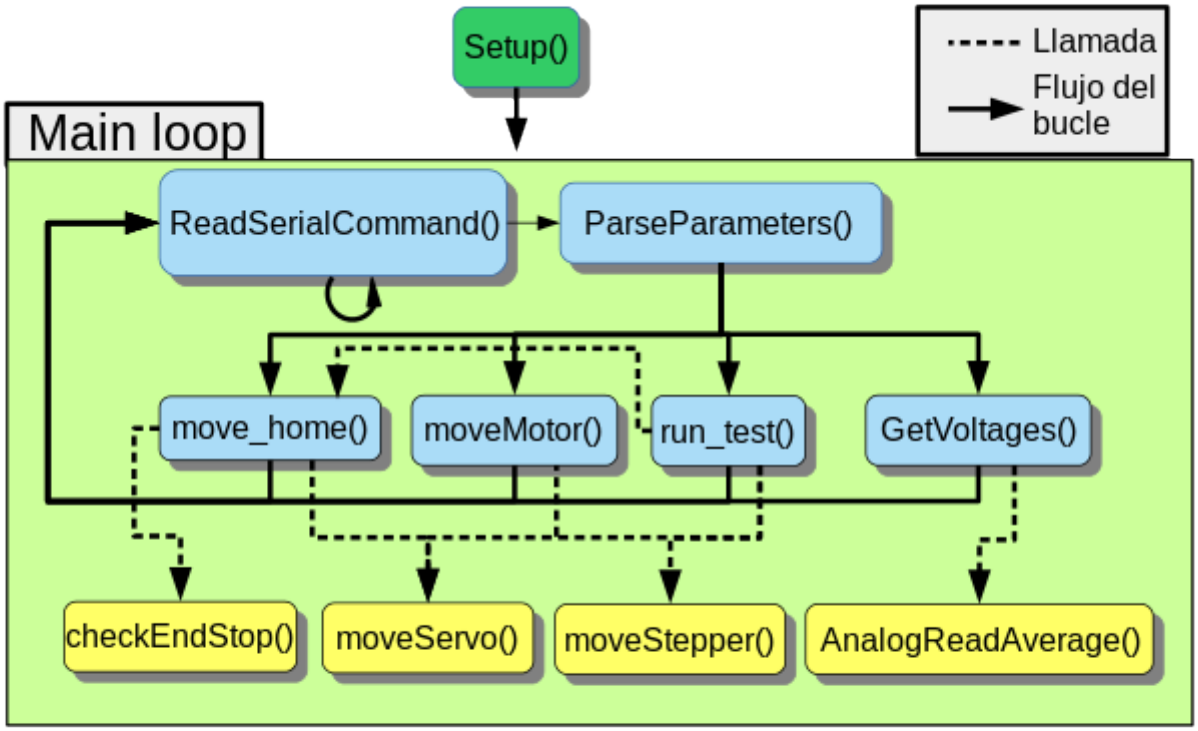


Figura 3.31: Diagrama funcional del firmware implementado para AquaRide. En azul, las funciones principales que componen cada ciclo del bucle, y en amarillo las que son llamadas múltiples veces en cada ciclo. El código fuente está disponible en el Anexo E: Código utilizado: Firmware para la placa Sanguinololu.

3.5.2.1 *Control de movimiento*

El control de movimiento del sensor está gestionado por seis funciones. De nuevo, el código de todo el firmware está disponible en el anexo E: "Código utilizado: Firmware para la placa Sanguinololu".

- "moveMotor": Esta es la función a la que se llama desde el bucle principal y sirve para encapsular a las dos funciones de bajo nivel para los distintos tipos de motor.

- "moveServo": Se encarga de gestionar la posición del servo y la velocidad de movimiento del mismo.

- "moveStepper": En este caso, la funcionalidad implementada es bastante más compleja que en el caso del movimiento de un servo, ya que se ha incluido:
  - Los ciclos de rotación de los motores paso a paso.
  - La conversión al esquema de movimiento CoreXY.
  - La limitación de movimiento vía sensores de fin de carrera.

- "move_home": Esta función calibra el sistema y establece los límites de movimiento. Para ello, utiliza las funciones moveStepper y moveServo con un flag de calibración que permite utilizar los detectores de fin de carrera como límites del movimiento, en lugar de la limitación virtual establecida habitualmente. Esta calibración sitúa el origen de el área virtual que se puede recorrer en la posición real de origen.

- "CheckEndStop":Esta función es llamada por "move_home" y se limita a comprobar el valor del detector de fin de carrera que se haya indicado como argumento de entrada. Devuelve un flag que al estar activado detiene el movimiento del sistema.

- "run_test": Esta función se encarga de realizar un proceso de medida de la precisión del sistema. Para ello, realiza una serie de movimientos alrededor de la pecera, y al terminar llama a la función "move_home" en un modo especial, que al terminar muestra el número de pasos de desvío con respecto al caso teórico.
  - Las medidas realizadas en la versión de AquaRide construida en este proyecto, dan como resultado una precisión de 0.2%, 0.5% y 0,03% para los ejes X, Y y Z cada 90.000, 100.000 y 60.000 pasos respectivamente.

Estas funciones se limitan a transformar comandos de movimiento recibidos en los comandos necesarios para mover los servomotores y motores paso a paso. Son funciones de nivel bajo que necesitan un nivel de control por encima para realizar tareas como mapeados o exploración. Esta labor de control se realiza en el ordenador conectado a la placa, que va indicando los pasos a seguir de uno en uno. Esto permite variar rápidamente los algoritmos de búsqueda sin necesidad de cambiar el firmware cargado en la placa, que permanece estático.

3.5.2.2 Adquisición de datos

La medida de voltajes en el sensor se realiza mediante el uso de dos funciones propias:

- "GetVoltages". Llamada al enviar el comando por puerto serie "sample N". Este comando indica a la placa que ha de realizar una ronda de N muestras para cada uno de los electrodos y combinaciones pre-programadas. La función gestiona la realización de medidas y la gestión de los tiempos para el adecuado muestreo de los voltajes. Gracias a los tiempos de espera incluidos en esta función, se puede simplificar el comportamiento

del medio como un comportamiento resistivo (Como se verá en el apartado 3.4.4: El modelo eléctrico)

También incluye la emisión de los voltajes adecuados por los electrodos. Tiene integrado por tanto el algoritmo explicado en la sección 3.4.3: El algoritmo de medida. Para obtener cada una de las cuatro medidas (V10, V20, V13 y V23), esta función llama a "AnalogReadAverage".

- "AnalogReadAverage". Esta función recibe el número de muestras a realizar y toda la tanda de medidas y calcula el valor medio de las mismas, eliminando el ruido existente en el sistema, tal y como se explica en la sección 3.4.2: Ruido en las medidas.

Para establecer los valores de salida adecuados en la emisión de pulsos y realizar las lecturas, se hace uso de las funciones nativas de Arduino "DigitalWrite" y "AnalogRead" en los pines A0 a A3 de la placa Sanguinololu (ver Anexo B: Conexiones y montaje). Para realizar los muestreos en los momentos oportunos, se ha utilizado la función también nativa "delay".

3.5.2.3 *Interfaz Sanguinololu-PC*

La conexión entre el programa desarrollado en Python, ejecutado en un ordenador y el firmware en la placa Sanguinololu se ha realizado a través del "Serial Port" (Ver diagrama en la Figura 3.32). Para poder utilizar la tarjeta en la adquisición de datos y el movimiento del sensor, es necesario un proceso controlador que gestione en un nivel superior los procedimientos de

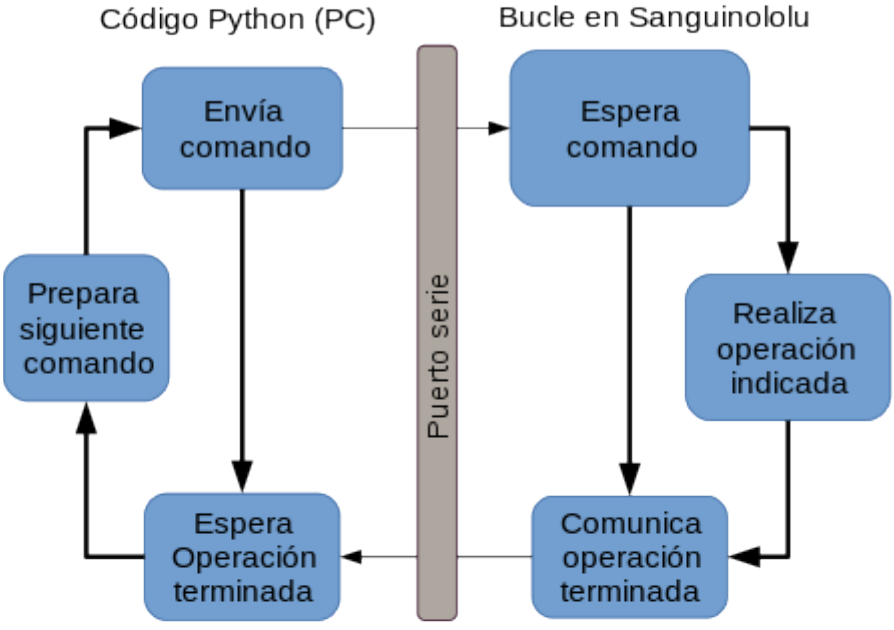


Figura 3.32: Diagrama de flujo de la interacción entre Python y la placa Sanguinololu

mapeado o exploración del terreno. Esta comunicación paso a paso requiere un pequeño protocolo como el que se ha desarrollado, presente en la Figura 3.33. Según este protocolo, es el proceso de Python el que comienza las tareas, enviando un comando al que la placa contesta opcionalmente con datos y obligatoriamente con un "IDLE" al final, que indica al proceso de control que la acción ha terminado de realizarse.

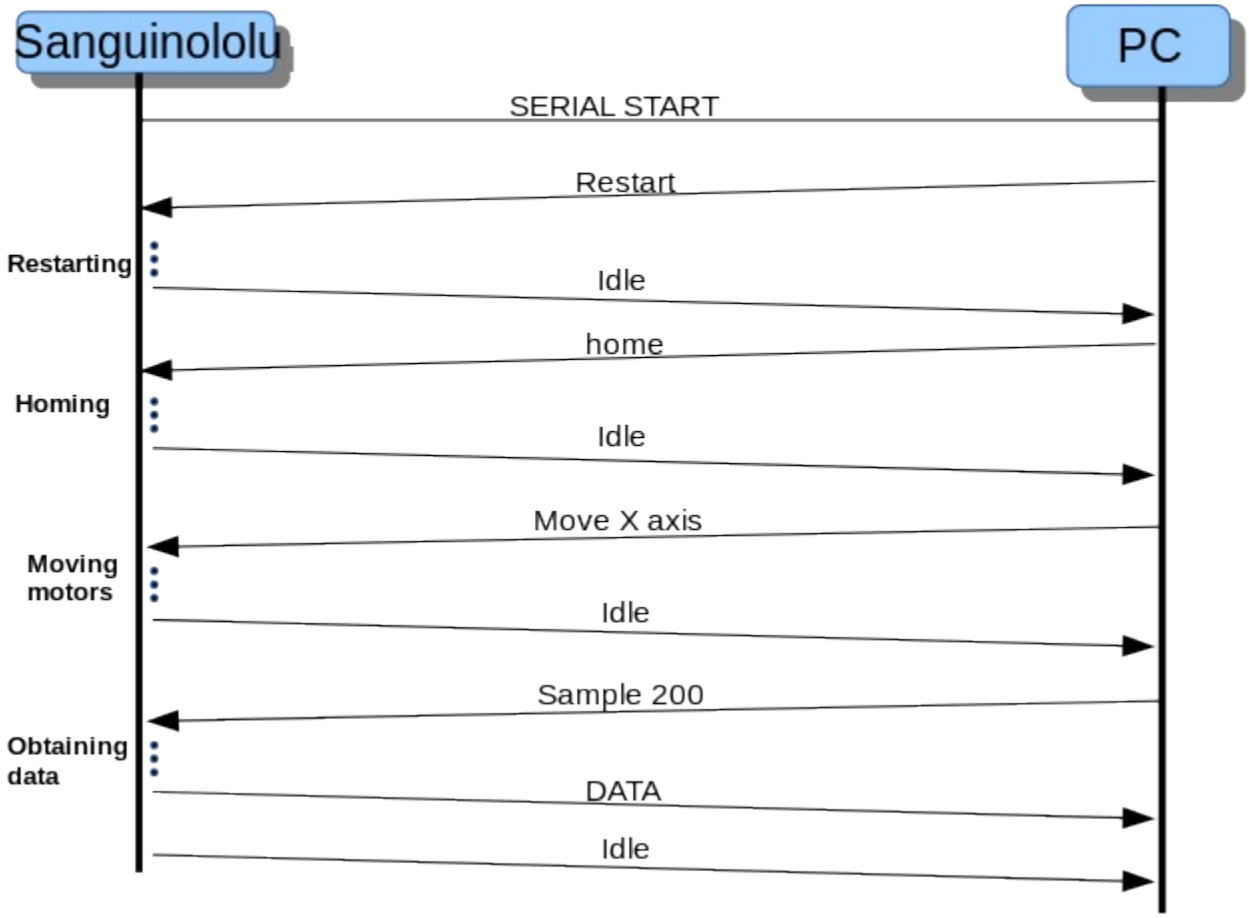


Figura 3.33: Detalle de la interfaz serie entre Sanguinololu y PC. Ejemplo de envío de comandos para la comunicación y realización de una operación de movimiento y otra de adquisición de datos.

Como notable problema en cuanto a la transmisión de datos, un limitante bastante importante en este aspecto es la velocidad de transmisión de datos que permite el puerto serie. Este problema destacó en una de las primeras versiones de AquaRide, en la que la eliminación de ruido realizando la media de muchas muestras era realizada en Python, y cada una de las muestras ruidosas se enviaba como dato a través de este puerto. Esto causaba saturación en la transmisión y era necesario introducir retardos para poder transmitir poco a poco todos estos datos.

Para solventar este problema y hacer el proceso de adquisición mucho más rápido, se decidió realizar la eliminación de ruido en la placa Sanguinololu en lugar de en el PC, de tal manera que para cada posición de medida, en la última versión AquaRide solo envía un único valor.
Es importante destacar que este cambio se realizó después de las conclusiones obtenidas en la sección 3.4.2: "Ruido en las medidas", que validaban este procedimiento.
Por este motivo, uno de los programas desarrollados encargado de realizar esta labor,"DataFilter.py" queda descartado ya que no es necesario su uso.

Cómo ultimo aspecto a destacar, posible mejora para este sistema es la utilización de un protocolo más complejo, que implemente más palabras de señalización y que permita llevar desde el controlador de AquaRide en Python un control más exhaustivo de los procesos que se están realizando, o un sistema de recuperación y detección de errores en la placa Sanguinololu.

3.6 Software: Control del sensor a alto nivel en Python.

De cara a la gestión a alto nivel de las tareas a llevar a cabo por AquaRide, es necesario crear un programa que envíe por puerto serie los comandos de movimiento, calibración y medida. Para realizar esto, se ha optado por utilizar como base el lenguaje Python, que ofrece una serie de librerías que facilitan todo este proceso. En concreto, las librerías más importantes que se han utilizado han sido:

- PySerial: Permite la comunicación a través de un puerto serie de manera sencilla.
- NumPy: Facilita el trabajo con matrices de datos.
- Math: Utilizada para realizar operaciones matemáticas poco habituales, como redondeos.

Gracias a estas librerías se han desarrollado los siguientes programas:

1. "Sample5K". Al ejecutar este código, se realiza una tanda de 5.000 muestras en la posición en la que esté actualmente el sensor. Este código ha sido el empleado para realizar el experimento visto en el apartado 4.1: Lectura de los electrodos en posición estática.
2. "Sweep.py". Este código realiza un barrido en línea del eje que se indique, dejando los otros 3 fijos en un determinado valor. Este código ha sido el empleado para realizar el experimento visto en el apartado 4.2: Barrido en línea.
3. "2D_Mapping.py". Similar al caso anterior, este programa realiza un barrido en un plano XY a la altura Z y ángulo A indicados. Este código ha sido el empleado para realizar el experimento visto en el apartado 4.3: Mallado de un plano horizontal.

4. "2D_Explore", que realiza una tarea similar a "2D_Mapping.py" pero en este caso analizando los valores recibidos para detectar objetos en su trayectoria. Este código ha sido el empleado para realizar el experimento visto en el apartado 4.4: Detección de objetos en plano horizontal.
5. "Histogram.py", que realiza un análisis de las propiedades estadísticas de la muestra provista.
6. "CsvPlotter.py". Representa las curvas de los voltajes guardados en los ficheros ".csv".
7. "CsvPlotter2D.py". Realiza la misma función que "CsvPlotter.py" pero representando los datos de barridos en un plano de dos dimensiones. Estas representaciones se hacen en formato imagen.

En el capítulo 4 "Experimentos" se muestran las gráficas obtenidas usando estos programas, cuyo código fuente se puede consultar en la sección de anexos (G Código utilizado: Python).

4 Experimentos

En esta sección se hace un repaso de los experimentos que se han realizado durante la construcción de AquaRide y una vez estaba finalizado. Antes no obstante de comenzar con los resultados de los experimentos, se va a explicar un aspecto práctico importante.

Debido a que las piezas del sistema ocupan espacio, el área navegable de la pecera no son exactamente 44.3x44.3 cm en el plano XY (ver Figura 4.1), sino que se le ha de restar una distancia debida a las dimensiones de las distintas piezas.

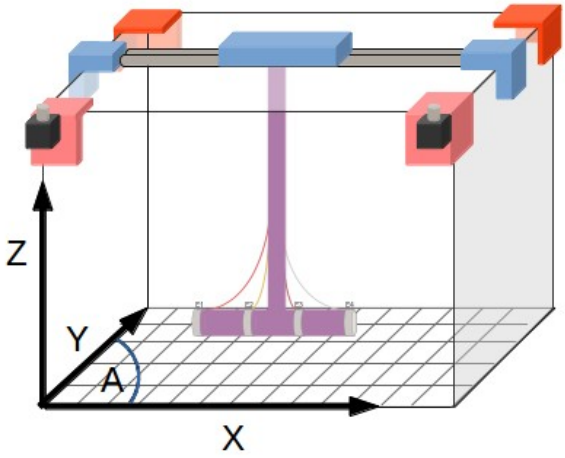


Figura 4.1: Situación de los ejes con respecto a la pecera.

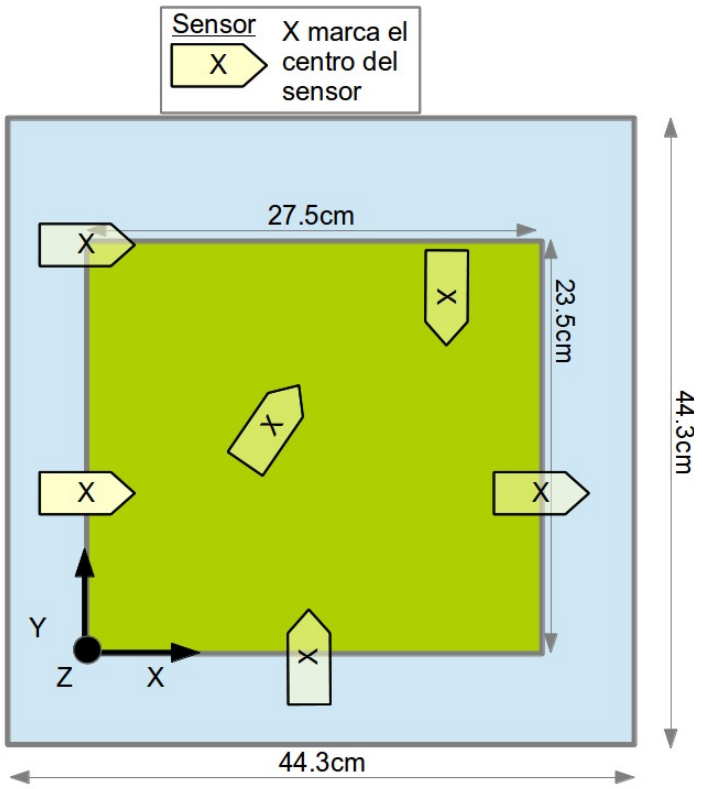


Figura 4.2: Superficie navegable real del acuario en vista cenital (área en verde). Esto es debido a la posición y anchura de algunas de las piezas que componen el sistema. Es inevitable dado que muchas de las piezas tienen un grosor que viene determinado por su función.

Además, también es importante tener en cuenta que los obstáculos tienen un efecto mayor en la superficie navegable, debido a las dimensiones de la sonda. Este efecto se aprecia de mejor forma en los experimentos finales de detección de objetos.

En la Figura 4.2 se puede ver cuáles son las dimensiones de navegabilidad reales de la pecera (27.5cm x 23.5cm).

Este fenómeno también ocurre en el eje Z, pero apenas es significativo.

4.1 Lectura de los electrodos en posición estática

Este primer experimento se realizó para verificar el funcionamiento de la parte de adquisición de datos del sistema, es decir:

- La conexión entre el sensor y los pines analógicos de la placa Sanguinololu.
- El tratamiento de los datos por el firmware desarrollado.
- El envío de los datos por el puerto serial del ordenador, incluyendo el protocolo de paso de datos.
- La recepción y guardado de datos en formato ".csv" por parte del código Python.
- La representación de los datos.

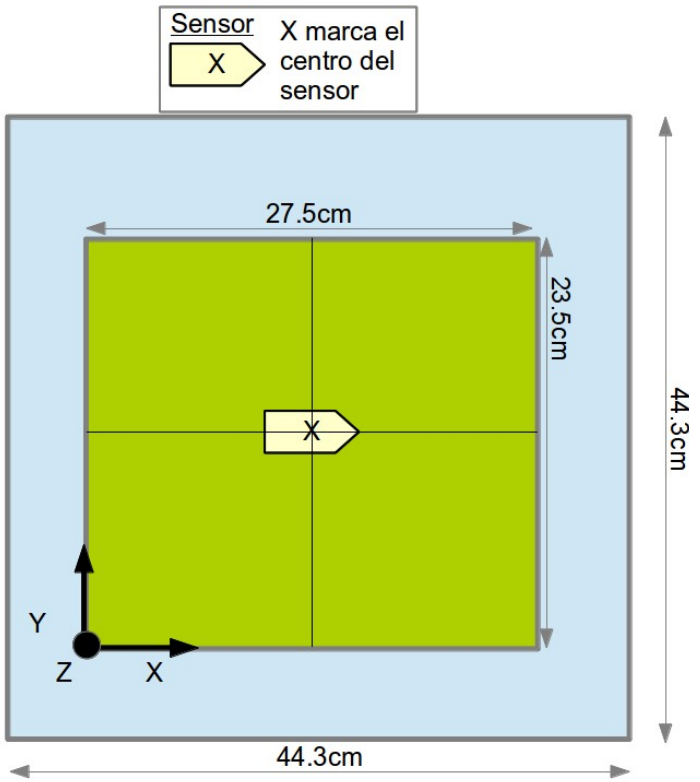


Figura 4.3: Diagrama de experimento para lectura estática del medio.

Para ello se situó el sensor (El sistema CoreXY de desplazamiento ya está perfectamente calibrado y en funcionamiento en este punto del proyecto) en el centro del acuario y se hizo una extensa tanda de medidas. Debido a que los datos obtenidos fueron los utilizados para caracterizar y eliminar el ruido, en estos datos todavía no se ha aplicado automáticamente el filtrado del mismo. En la Figura 4.22 se puede ver el diagrama del experimento y en la Figura 4.4 los resultados ya vistos del mismo. El código ejecutado en este experimento es "Sample5K.py".

4.2 Barrido en línea

En este caso (Figura 4.4), el experimento consiste en establecer fijos los ejes Y, Z y A de la pecera y desplazar el sensor en el eje X, trazando una linea recta virtual y haciendo medidas cada cierta distancia. Tanto las posiciones de Y, Z y A, así como el número de medidas realizadas en el barrido y el número de posiciones a lo largo del eje X se pueden establecer modificando el código ejecutado "Sweep.py"

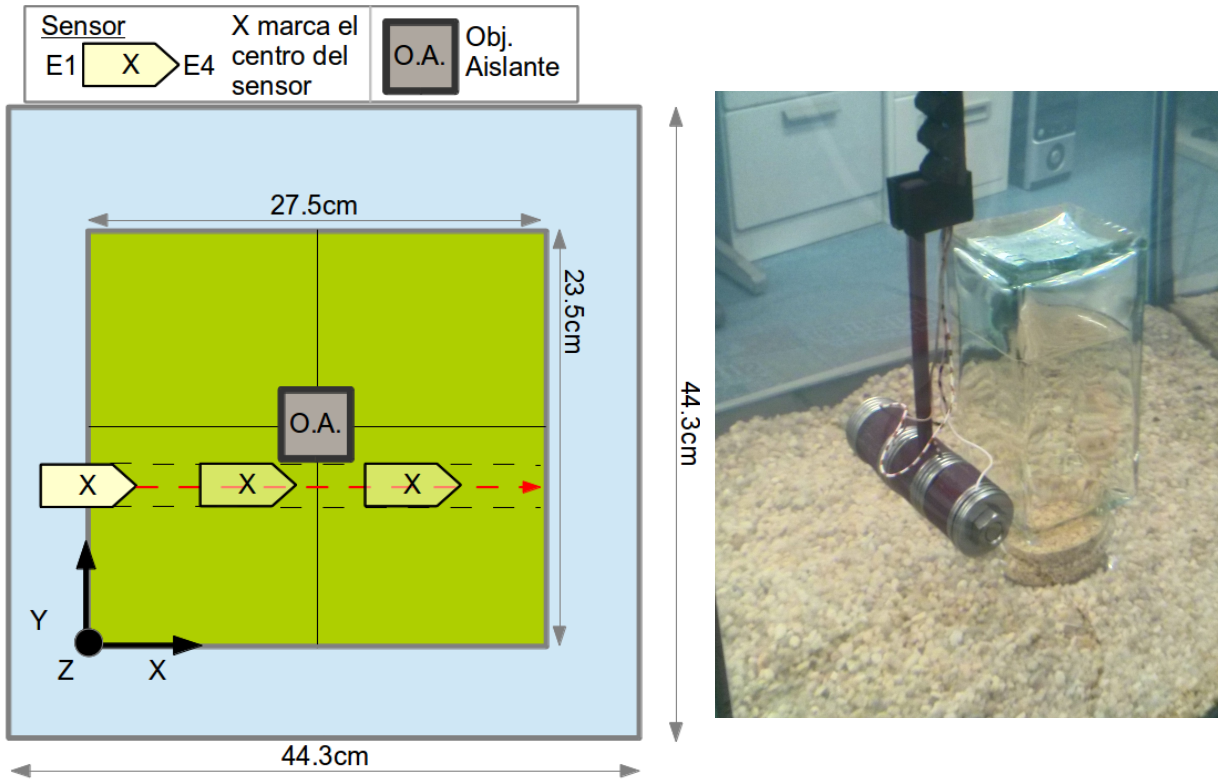


Figura 4.4: A la izquierda, diagrama del experimento de barrido lateral. A la derecha, foto del experimento durante su realización (con un objeto aislante). En el primero de los dos experimentos el objeto aislante se retira.

4.2.1 Pecera vacía

La primera prueba (Figura 4.5), que sirve en este caso como baseline para realizar comparaciones con los siguientes experimentos, tiene como resultado el que se muestra en la Figura 4.6. Las medidas, excluyendo los efectos de borde, son estables a lo largo del recorrido.

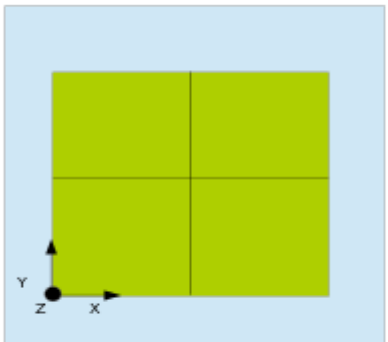


Figura 4.5: Estado de la pecera en esta primera parte del experimento: Vacia.

En esta figura se ve por primera vez cómo se ha gestionado la representación de los datos en cuanto a las coordenadas. Tal y como está preparado, el sistema siempre realiza el recorrido completo desde un lateral de la pecera hasta el otro, y divide el número de medidas solicitado distribuyéndolo en los segmentos necesarios. Por ejemplo, un experimento en el que se soliciten tres medidas realizaría dos medidas en los extremos y una en el centro de la pecera. Por este motivo, en las gráficas (tanto en 1D como en 2D) siempre se muestra un experimento que recorre la pecera completa, y lo único que varía es la resolución, es decir, el número de sitios en los que se mide. Esto hace que las curvas de las gráficas sean más suaves a mayor número de puntos, aunque es el parámetro que más afecta al tiempo que se tarda en realizar los experimentos.

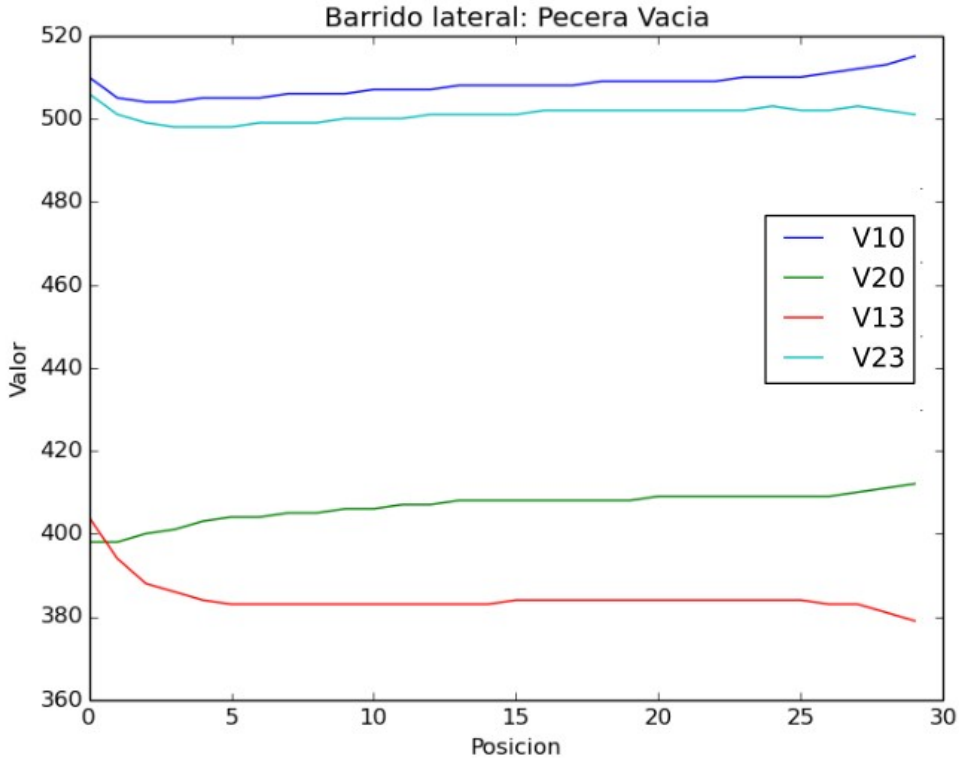


Figura 4.6: Barrido en eje X sobre una pecera vacía. Las medidas, excluyendo los efectos de borde, son estables a lo largo del recorrido.

4.2.2 Pasada lateral a un objeto aislante

En este caso, se sitúa un bote de cristal aislante paralelo a la trayectoria del sensor (Figura 4.7). Para ver claramente los efectos de su presencia, se ha situado a una cercana distancia de aproximadamente 1 cm.

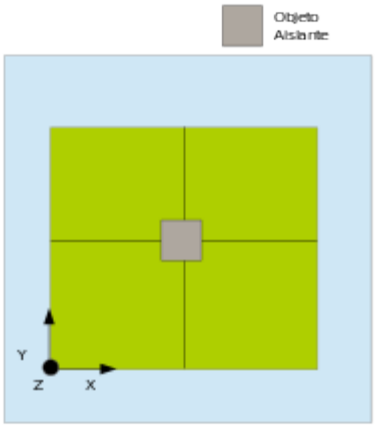


Figura 4.7: Estado de la pecera en esta fase del experimento. Se sitúa un objeto aislante en el centro.

Como se puede ver en la Figura 4.9, el efecto introducido por el elemento aislante durante el recorrido del sensor es similar a introducir una resistencia desplazante en el esquema visto del divisor de tensión (ver Figura 4.8 y Figura 4.9). Una representación más específica de esta casuística se puede ver en la Figura 4.8.

La presencia de la resistencia externa en la zona entre el electrodo medido y el electrodo a 5V produce una caída en la medida y la presencia de la resistencia externa en la zona entre el electrodo a 0V y el electrodo medido produce un aumento en el voltaje medido. No obstante, este efecto no es discreto, sino que su efecto varía conforme a la posición en la que se encuentra de hecho esto se ve claramente en la Figura 4.9, ya que hay un decaimiento lineal entre los máximos y mínimos de cada función.

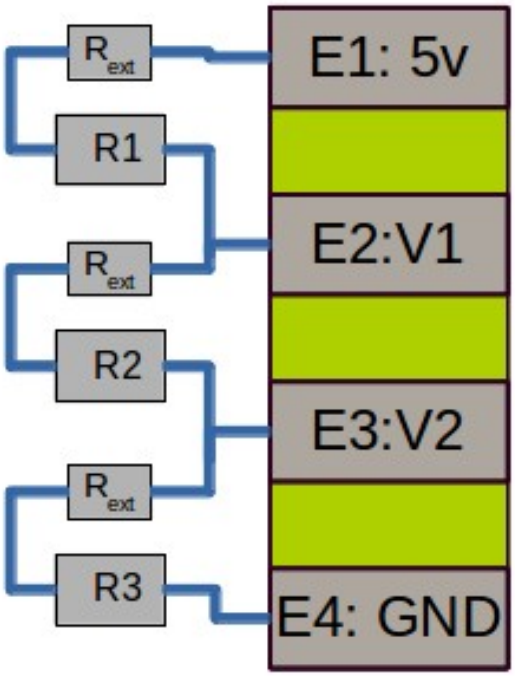


Figura 4.8: Esquema simplificado del efecto de un objeto externo aislante cerca del sensor. R1, R2 y R3 representan las resistencias típicas existentes en un medio acuático vacío de elementos externos, y sólo varían con respecto a la resistividad del agua.Rext representa un elemento aislante. En caso de introducir un elemento más conductivo que el agua, como un elemento metálico, es posible modelar el comportamiento situando Rext como una resistencia en paralelo a R1,R2 o R3, dependiendo de su posición.

Estos dos puntos proporcionan información en cuanto a la posición del objeto aislante, pero hay otro punto en el que también se puede extraer información: la posición en la que el voltaje no es alterado, que representa que la resistencia a efectos prácticos está situada en paralelo al electrodo, compensándose los efectos resistivos por encima y por debajo del electrodo medido.

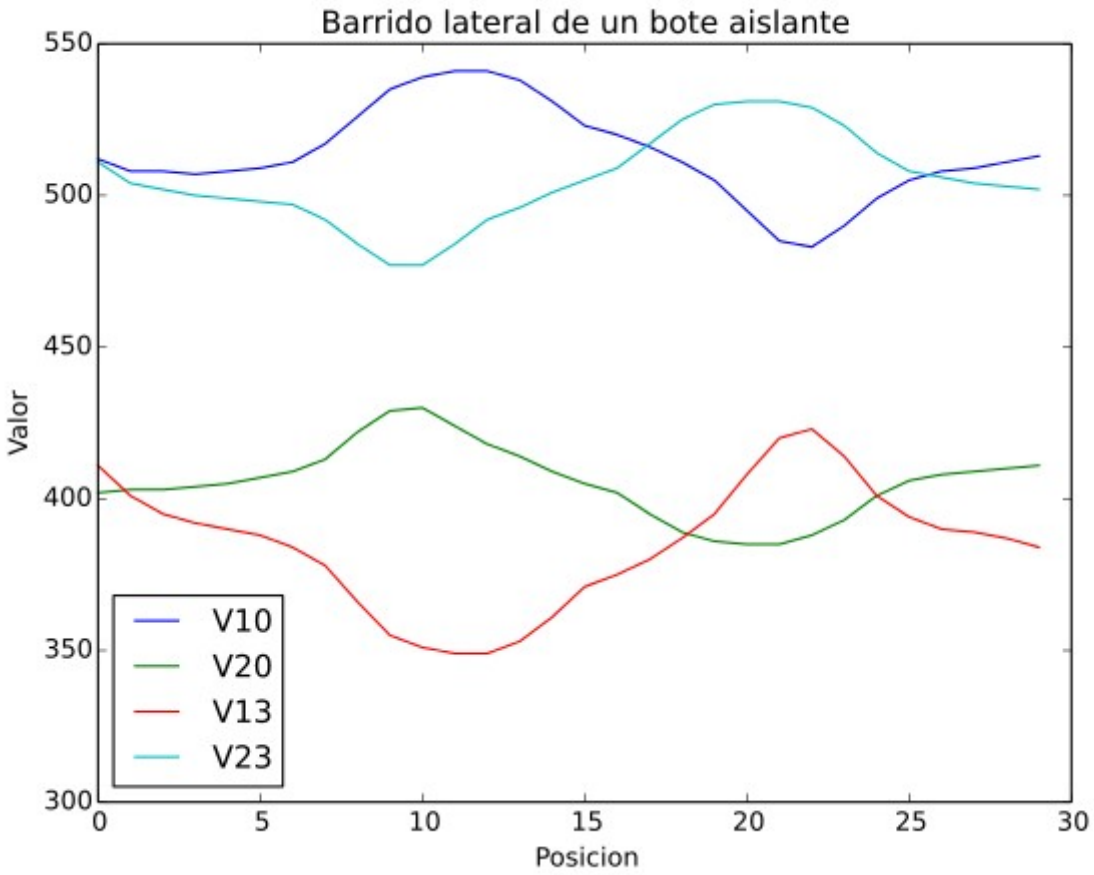


Figura 4.9: Barrido lateral a aproximadamente 1cm de una botella de vidrio cuadrada aislante situada en el centro de la pecera. Como se puede ver hay dos tipos de gráficas complementarias que corresponden a los mismos fenómenos pero midiendo el dipolo con la polaridad inversa.Las comparaciones más razonables se pueden realizar por tanto entre las gráficas dos a dos V10 y V13 por un lado y las gráficas V20 y V23 por otro. En estos casos se puede ver que los comportamientos en un mismo electrodo son consistentes entre distintas estimulaciones, al margen de efectos geométricos fruto del diseño del sensor.
En el caso de querer extraer información de manera cruzada entre el resto de gráficas, la información vital corresponde a los máximos y mínimos detectados por cada función de manera independiente, ya que debido a factores como el offset introducido por la distancia entre electrodos o la posición relativa del electrodo con respecto al centro del sensor, los electrodos no miden cosas gráficamente comparables.
Por último, se puede observar en menor medida que existe un efecto de bordes no despreciable debido a que las paredes del acuario son muros aislantes. Gráfica obtenida utilizando el código "CsvPlotter.py".

4.2.3 Conclusiones extraídas del experimento

Se puede comprobar que el sistema es capaz de detectar, con un grado de sensibilidad alto, la presencia de objetos en una distancia próxima al sensor, de aproximadamente 1cm. Por otro lado,

se ha constatado que el sensor permanece estable ante la ausencia de objetos modificadores del entorno.

4.3 Mallado de un plano horizontal

En este caso la prueba realizada es una pasada para un mismo Z por todo el plano XY de la pecera como el mostrado en la Figura 4.10. La altura Z de medida es aproximadamente 1cm por encima de la altura máxima del objeto aislante. Tanto las posiciones de Z y A, así como el número de medidas realizadas en el barrido y el número de posiciones a lo largo del eje X e Y se pueden establecer modificando el código ejecutado "2D_Mapping.py"

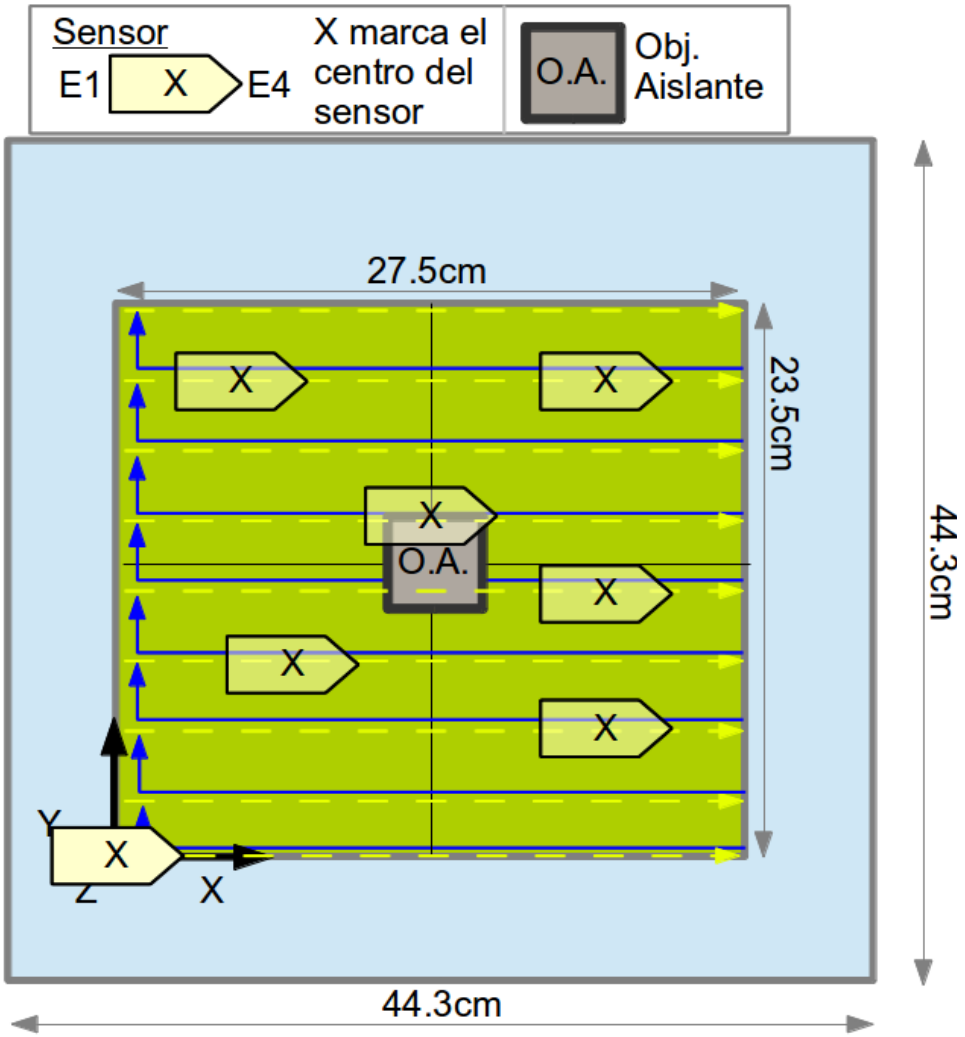


Figura 4.10: Diagrama del experimento de medida en todo el plano XY para una Z y A fijas. Variación de las posiciones, medidas en pasos de los motores, para los ejes X e Y durante el barrido en el plano XY. Para cada posición en el eje Y, se realiza un barrido en el eje X que vuelve al principio al terminar. En cuanto a la altura de Z, se hace a aproximadamente 1cm por encima de la altura del objeto aislante.

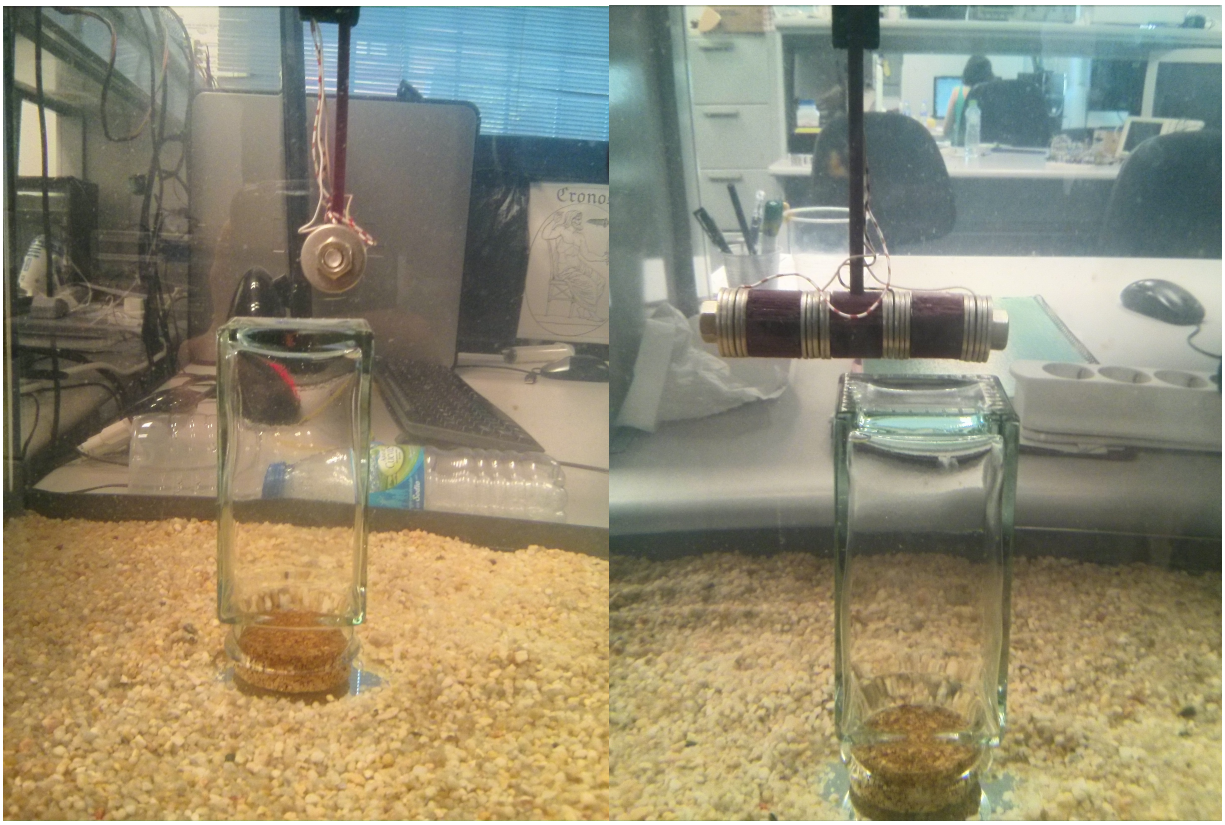


Figura 4.11: Fotografías del experimento de barrido superior en ejecución. La distancia en altura sobre el bote de vidrio es aproximadamente 1cm.

4.3.1 Pecera vacía

Por el mismo motivo que en el caso anterior, se realiza una prueba sobre el acuario vacío. En este caso, tal y como se ve en las Figuras 4.13, 4.14, 4.15 y 4.16, se reconoce claramente el efecto de bordes aparecido en las paredes del acuario, y cómo las aproximaciones frontal, trasera y lateral tienen combinaciones de efectos distintos en cada uno de los voltajes medidos.

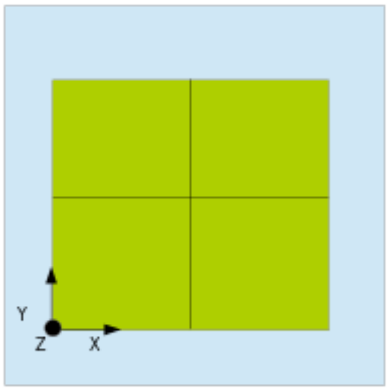


Figura 4.12: Estado de la pecera en esta primera parte del experimento: Vacía.

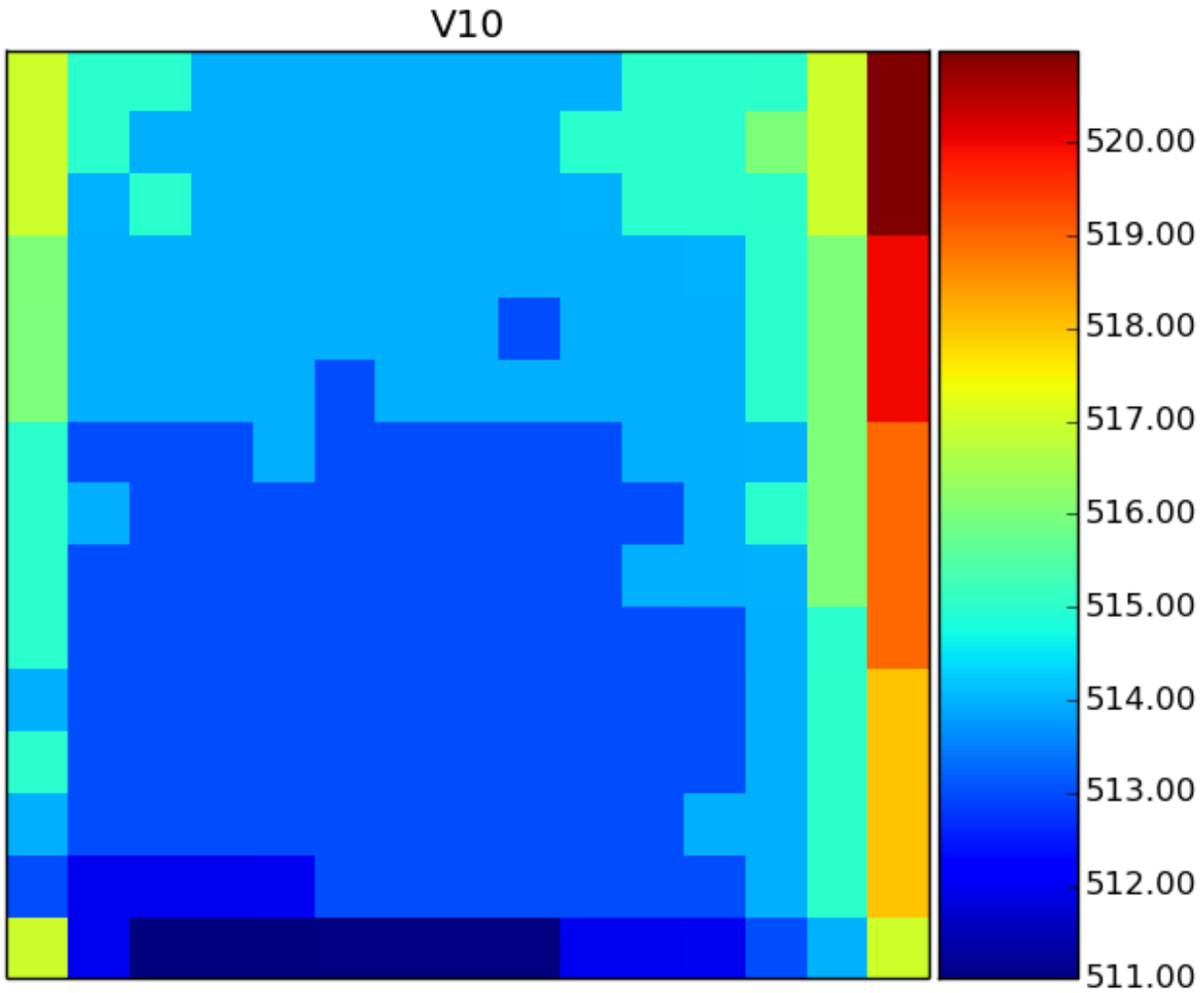


Figura 4.13: Resultado de la medida de una malla de 15x15 medidas en un plano XY a altura media de la pecera. Resultados para la medida V10. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

Se puede observar en estas figuras que efectivamente, no aparecen máximos ni mínimos locales correspondientes a objetos extraños, sino que la medida sólo varía en función del efecto de las paredes del acuario. Este comportamiento es similar para las cuatro medidas.

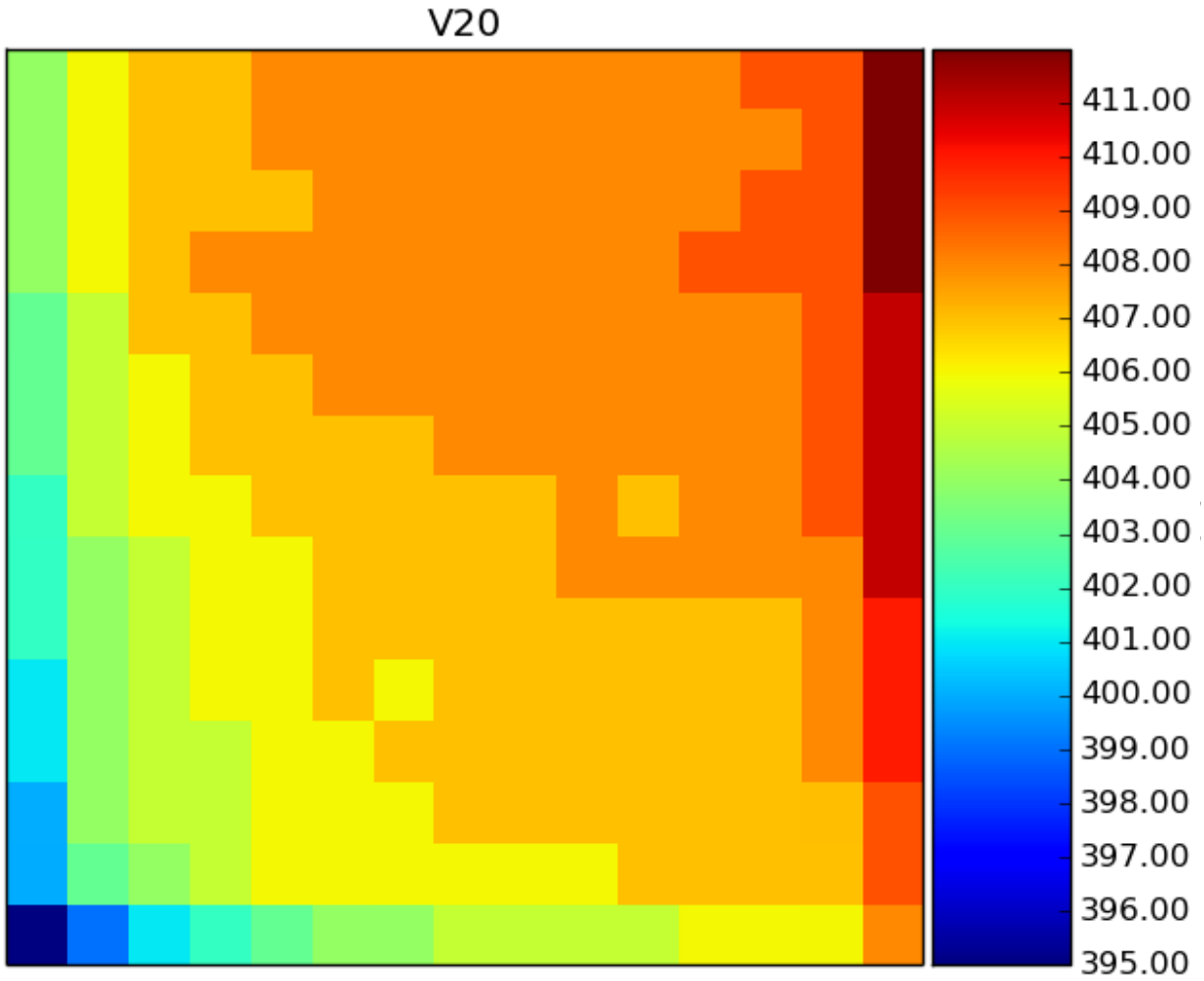


Figura 4.14: Resultado de la medida de una malla de 15x15 medidas en un plano XY a altura media de la pecera. Resultados para la medida V20. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v

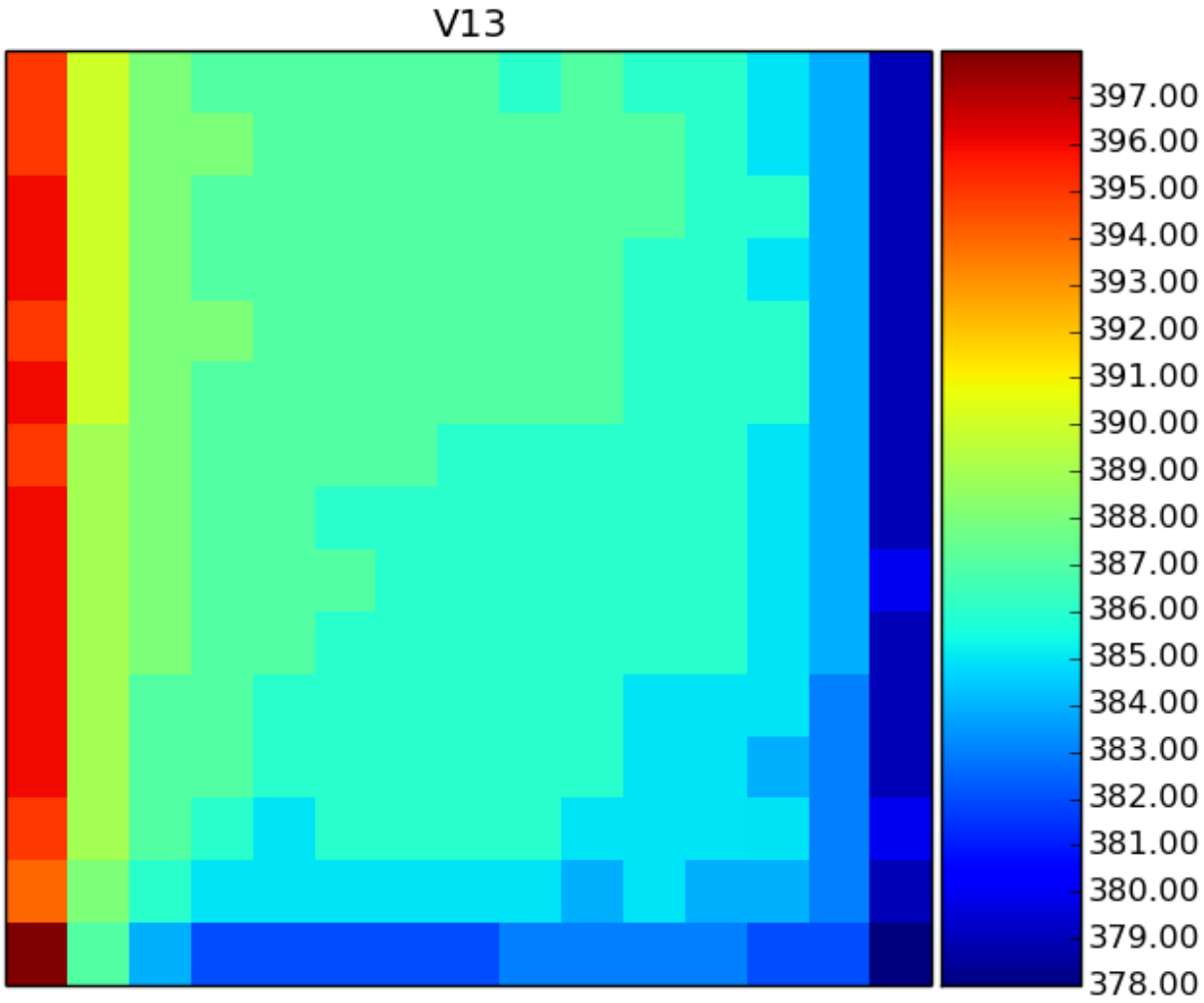


Figura 4.15: Resultado de la medida de una malla de 15x15 medidas en un plano XY a altura media de la pecera. Resultados para la medida V13. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v

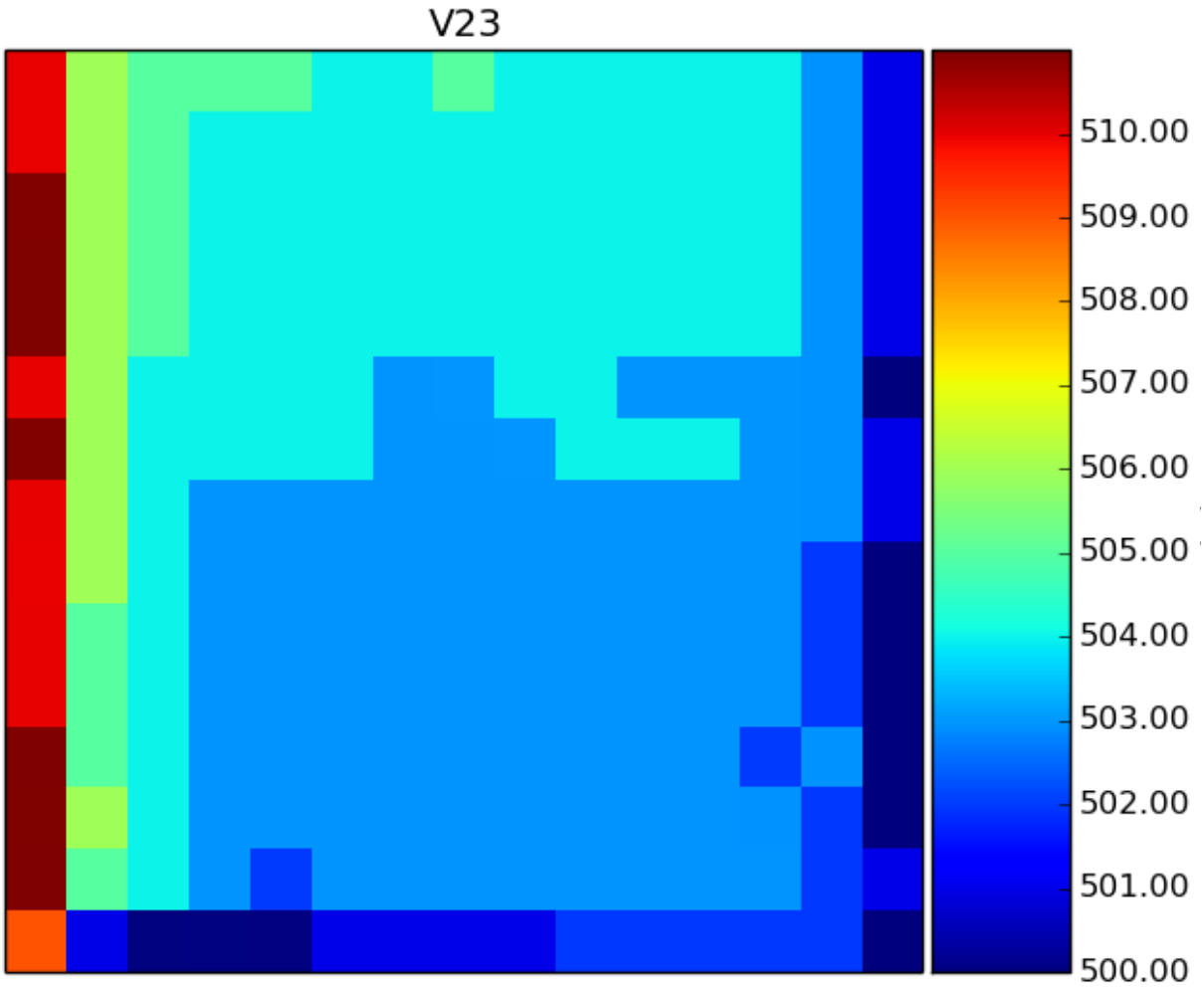


Figura 4.16: Resultado de la medida de una malla de 15x15 medidas en un plano XY a altura media de la pecera. Resultados para la medida V23. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

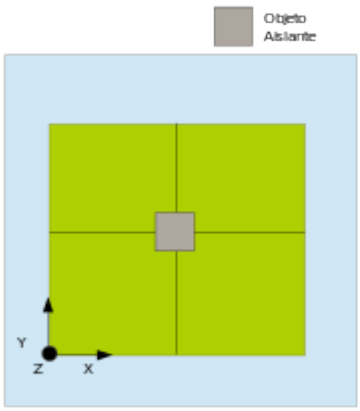


Figura 4.17: La pecera en esta fase del experimento. Se sitúa un objeto aislante en el centro.

4.3.2 Mallado del plano encima de un objeto aislante

En este caso, se realiza la misma prueba situando un bote aislante justo por debajo de la altura en la que se está realizando el mapeo (Figura 4.17). Se puede observar a continuación, en las Figuras 4.18, 4.19, 4.20 y 4.21 el contraste con respecto al caso anterior. Aparecen en todos los casos dos irregularidades en forma de máximos y mínimos que representan los dos puntos de máximo efecto del bote aislante. Estos puntos corresponden a los

máximos y mínimos detectados en el experimento visto en la Figura 4.9 y en este caso proporcionan mucha más información en cuanto a la posición del objeto.

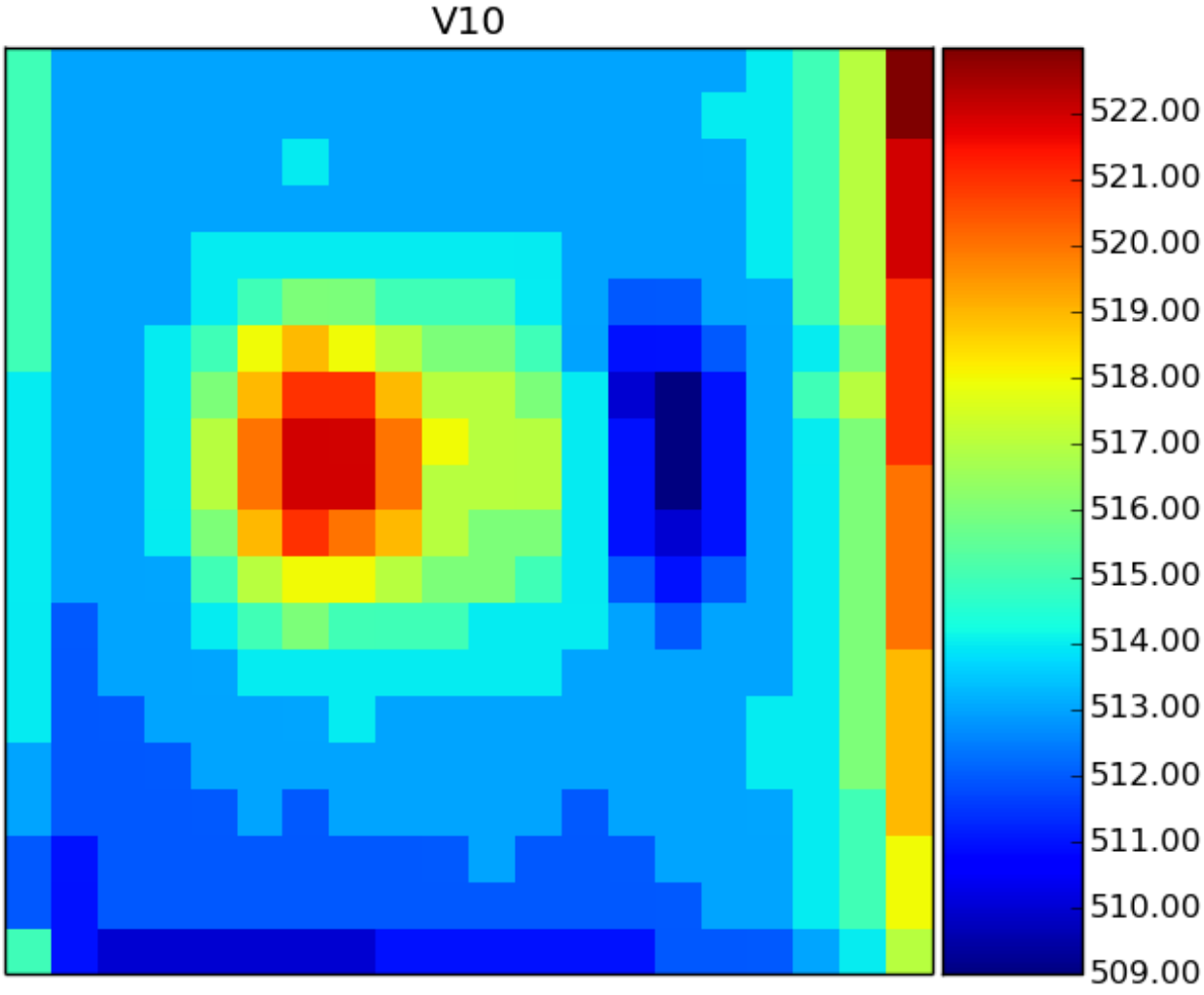


Figura 4.18: Resultado de la medida de una malla de 20x20 medidas en un plano XY por encima de un bote de cristal aislante. Resultados para V10. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

En este caso, aparecen las componentes resistivas propias del objeto introducido en la pecera. Un factor importante en la geometría de los resultados es la orientación del sensor, que coincide con la dirección del vector que une el máximo y mínimo aparecidos. En cuanto a la posición de los máximos y mínimos aparecidos, estas gráficas están sujetas de la misma forma que la Figura 4.9 a efectos como la posición relativa de los distintos electrodos al centro del sensor. Esto produce una diferencia entre las posiciones de tanto mínimos y máximos como de puntos intermedios "Ciegos". Esto permite, conocidas las posiciones de los electrodos, obtener más información

sobre la posición real del objeto cercano, ya que nunca se situará en una posición ciega para todos ellos.

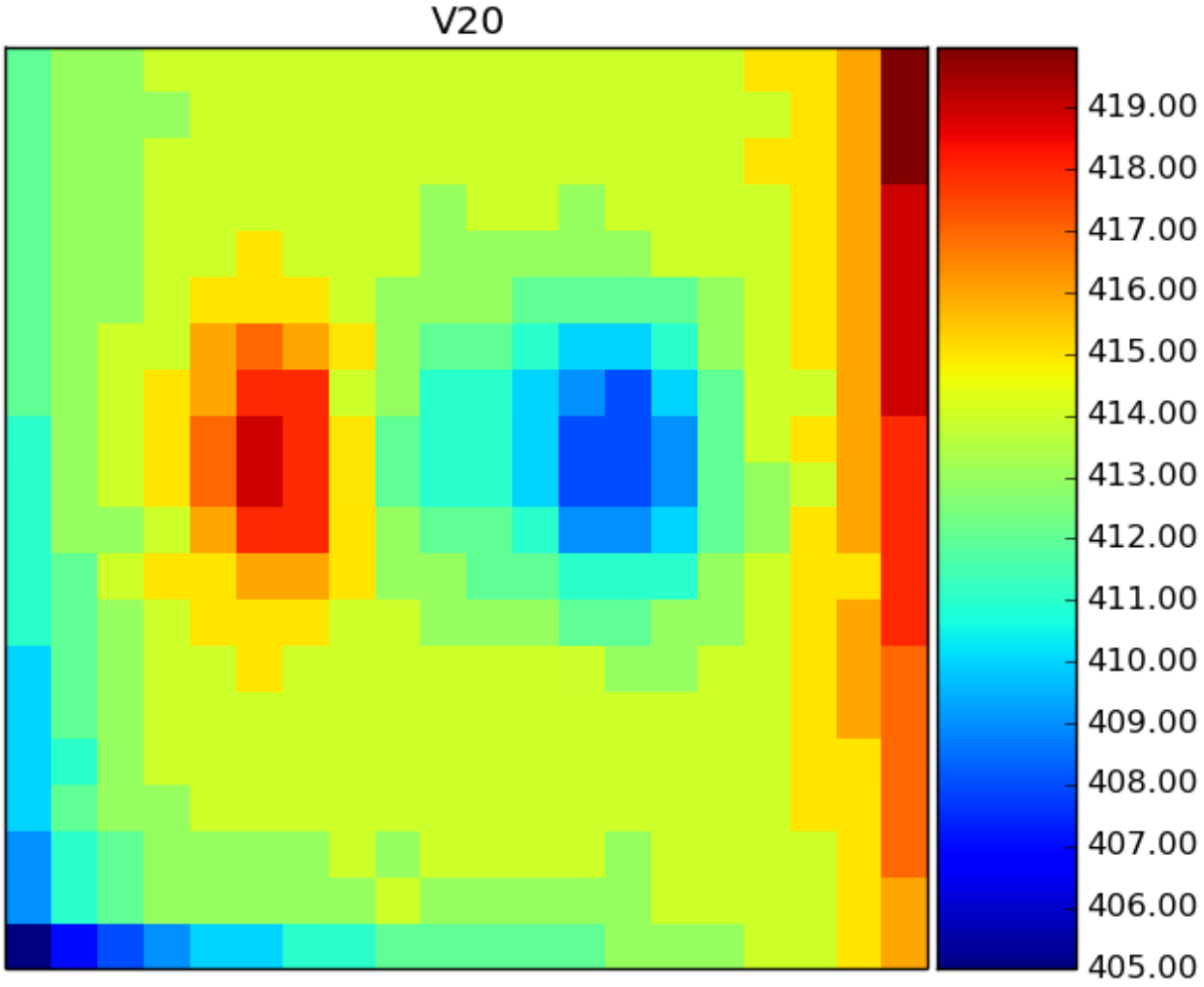


Figura 4.19: Resultado de la medida de una malla de 20x20 medidas en un plano XY por encima de un bote de cristal aislante. Resultados para V20. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

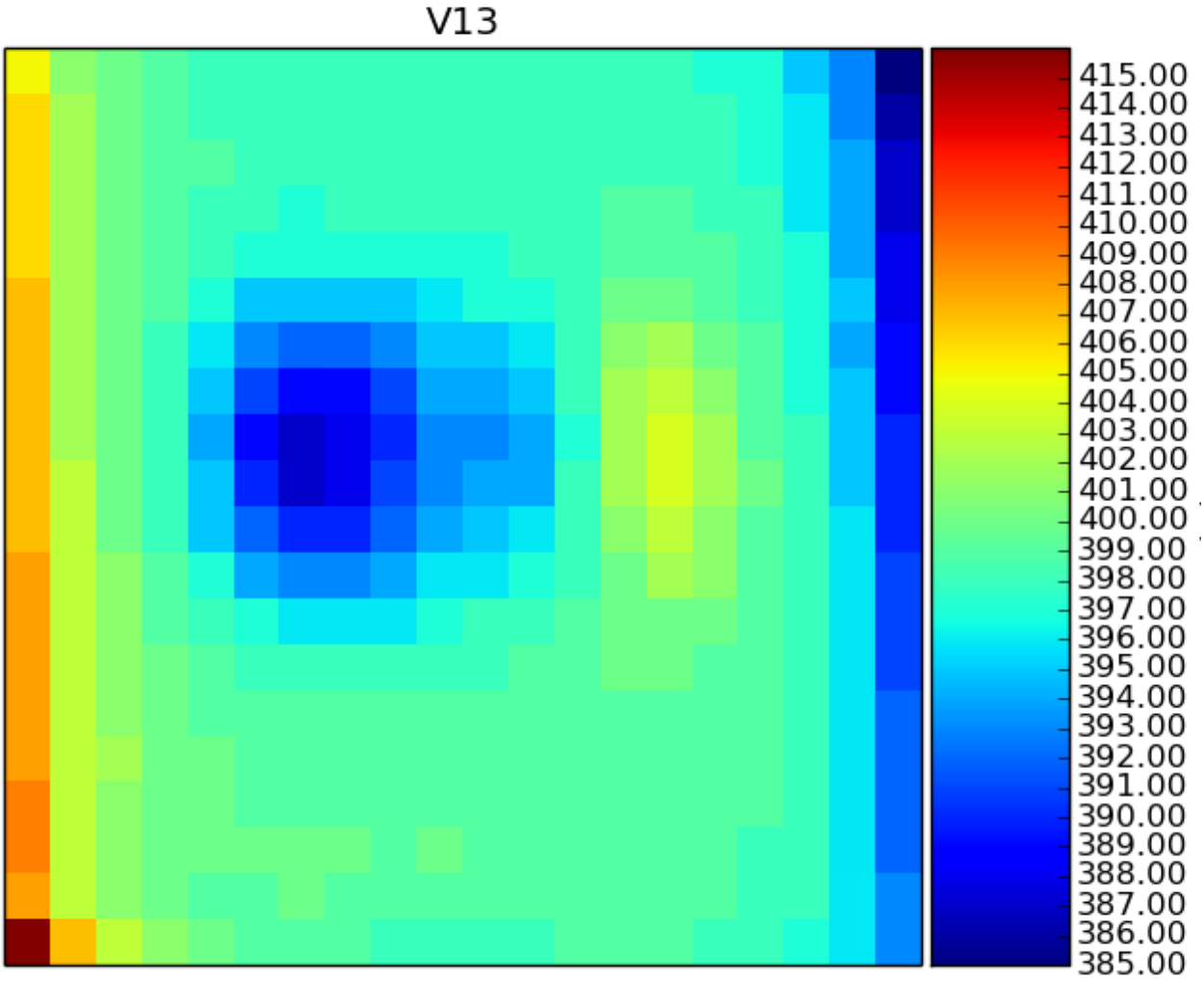


Figura 4.20: Resultado de la medida de una malla de 20x20 medidas en un plano XY por encima de un bote de cristal aislante. Resultados para V13. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

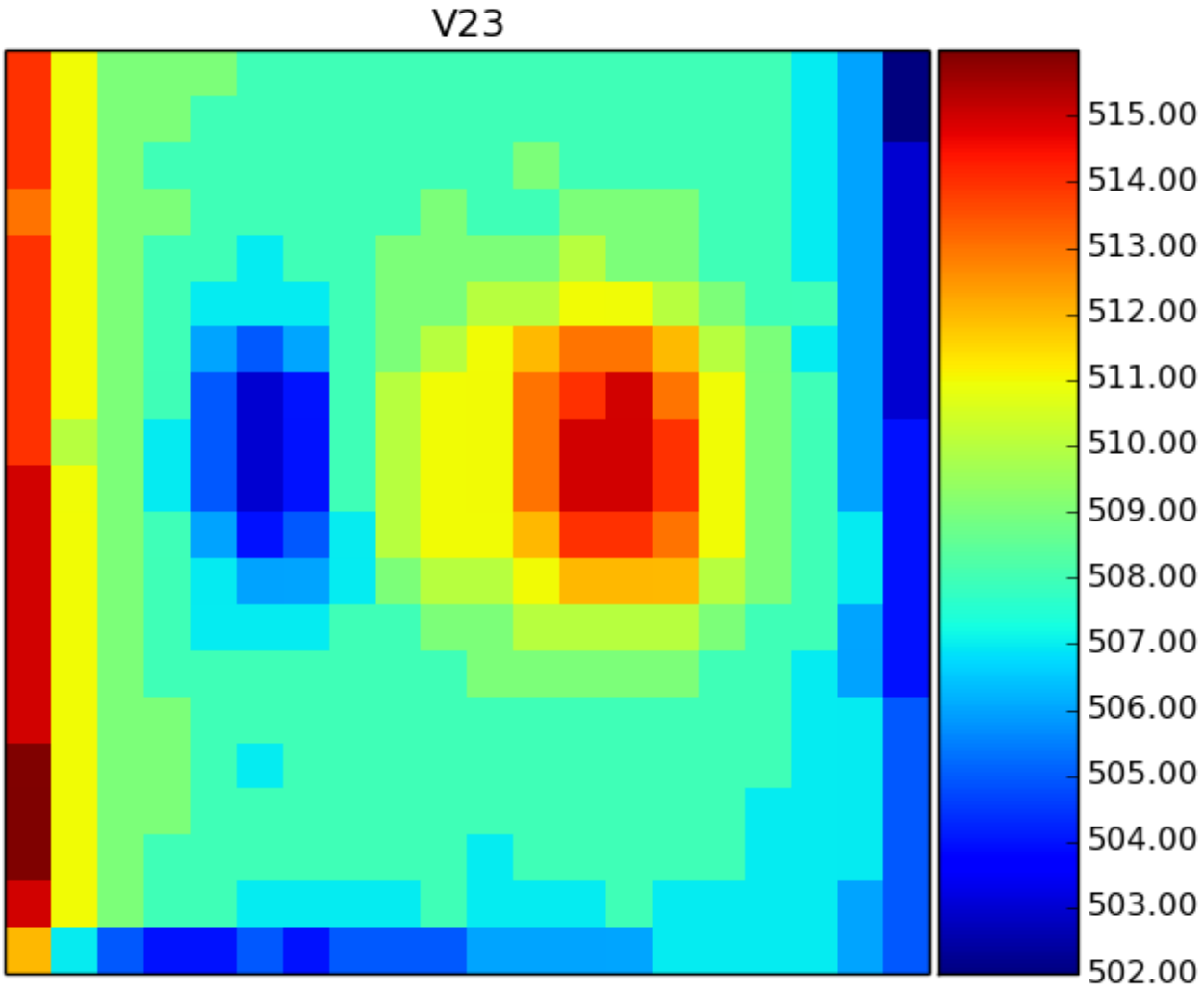


Figura 4.21: Resultado de la medida de una malla de 20x20 medidas en un plano XY por encima de un bote de cristal aislante. Resultados para V23.Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

4.3.3 Conclusiones extraídas de los experimentos de mallado en 2D

Tras este experimento, se concluye que el sistema es capaz de trabajar correctamente con mallados en 2D, tanto en la elaboración de los ficheros de datos como en la representación de los mismos. En cuanto a los datos obtenidos, vemos que el modelo del sensor es consistente, mostrándose cambios significativo en los mapas obtenidos, que permiten, como se va a ver a continuación, la implementación de algoritmos de localización.

4.4 Detección de objetos en plano horizontal

Tras comprobar con los experimentos anteriores que el modelo de comportamiento previsto para el sensor se cumple, se ha implementado un algoritmo en el código "2D_Explore.py" que permite detectar objetos y detenerse. Este experimento funciona de la misma manera que el experimento

de mapeo horizontal, pero con la condición de que si se detecta un obstáculo, se detiene el avance en esa dirección y se comienza de nuevo en la siguiente posición para el eje Y. Este nuevo comportamiento se puede ver en el diagrama de la Figura 4.22. Una fotografía del experimento durante su ejecución se puede ver en la Figura 4.23.

Tanto las posiciones de Z y A, así como el número de medidas realizadas en el barrido y el número de posiciones a lo largo del eje X e Y se pueden establecer modificando el código ejecutado "2D_Explore.py".

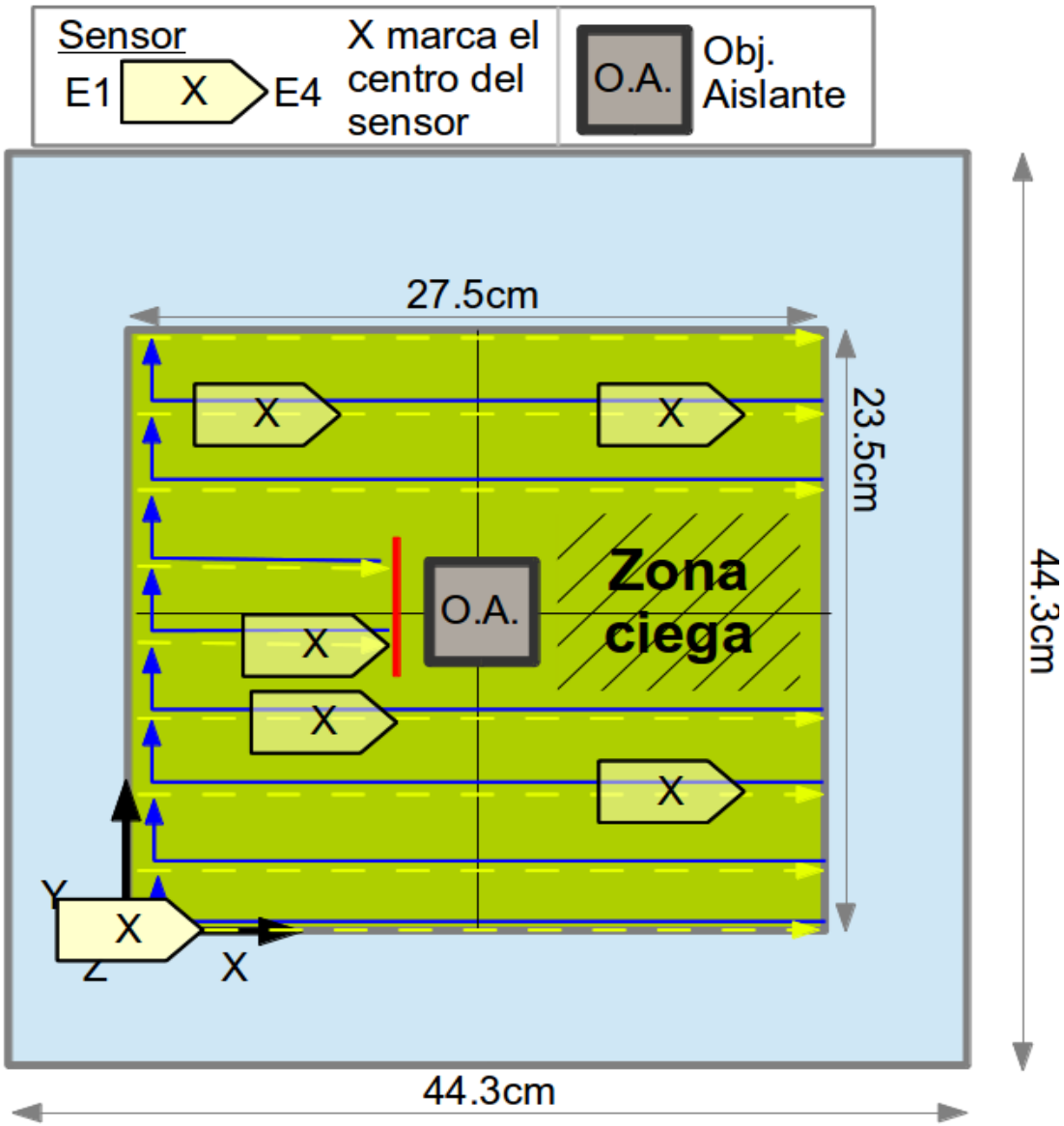


Figura 4.22: Diagrama del experimento de medida en todo el plano XY para una Z y A fijas, implementando el algoritmo de detección de objetos. Para cada posición en el eje Y, se realiza un barrido en el eje X que vuelve al principio al terminar o, en el caso de aparición de un objeto, se retrocede prematuramente. En cuanto a la altura de Z, se hace a mitad de la altura del objeto aislante para maximizar su efecto resistivo en las medidas.

Figura 4.23: Fotografía durante el proceso de barrido detectando objetos. En el caso fotografiado se está realizando el experimento con dos objetos incluidos: el bote utilizado en el resto de pruebas y un filtro de agua típico de peceras, que se puso para comprobar la robustez del sistema ante elements de naturaleza distinta.

El algoritmo de detección se basa en el comportamiento teórico y observado y utiliza la detección del cambio brusco de valores entre una medida y la siguiente. Para acentuar este efecto, se ha calculado un valor de "score" que acumula los cambios en cada uno de los electrodos y tipos de estimulación (Se utilizan V10,V20,V13 y V23, ver Tabla 3). Debido a que no todas las medidas se comportan igual, dependiendo de la estimulación realizada los valores de cambio afectan a la función de "score" de distinta manera.

Experimentos

Para un caso de aproximación por E4 como el que se ha estado haciendo hasta ahora, las medidas recogidas:

- V23 y V13 se hacen más pequeñas cuanto más cerca.
- V20 y V10 crecen al acercarse.

Como se puede observar en la Figura 4.24, el código implementado interpreta tal y como dice el modelo un cambio positivo en las medidas V10 y V20 y negativo en V13 y V23 como una aproximación a un objeto por delante. Sin embargo, un comportamiento inverso querría decir que el objeto se aproxima por detrás.

El hecho de utilizar tan sólo la medida anterior para calcular este "score" no es problemático ya que los cambios son exponenciales con respecto a la cercanía del objeto. No obstante, una mejora en este diseño puede ser la implementación de un mejor algoritmo de aproximación que tenga en cuenta más factores, como el cambio gradual en los anteriores pasos y detecte tendencias. Esto permitiría otras mejoras, como el acercamiento cuidadoso a un posible objeto desde una distancia mayor.

```
V10dif=lineSamples[-1,4]-lineSamples[-2,4]
V20dif=lineSamples[-1,5]-lineSamples[-2,5]
V13dif=lineSamples[-1,6]-lineSamples[-2,6]
V23dif=lineSamples[-1,7]-lineSamples[-2,7]

FrontIndicator= V10dif+V20dif-V13dif-V23dif
BackIndicator= -V10dif-V20dif+V13dif+V23dif


FrontFlag= FrontIndicator>objectThreshold and V10dif>0 and V20dif>0 and V13dif<0 and V23dif<0
BackFlag= BackIndicator>objectThreshold and V10dif<0 and V20dif<0 and V13dif>0 and V23dif>0
```

Figura 4.24: Cálculo de los indicadores de presencia en "2D_Explore.py". VXXdif representa la variación de esta medida para un cambio de posición. FrontIndicator y BackIndicator son las funciones aplicadas de "scoring". La bandera se activa si

Debido a que este algoritmo no está pensado para otras situaciones como por ejemplo la aproximación lateral, se ha añadido una bandera para indicar que todos las medidas individualmente se comportan de la manera esperada. Esto previene falsos positivos en situaciones no adecuadas, como las producidas en las esquinas del acuario.

Finalmente, el código interpreta que se ha encontrado un objeto cuando se activa la bandera de frente y se supera un cierto umbral en la puntuación. A continuación se muestra los resultados de realizar distintos experimentos implementando este algoritmo.

4.4.1 Detectando una pecera sin objetos

En un primer caso, para comprobar que el algoritmo no tiene falsos positivos en cuanto a la detección de objetos, se ha realizado un barrido en plano horizontal de una pecera sin ningún objeto aislante. Se puede observar a continuación, en las Figuras 4.26, 4.27, 4.28 y 4.29 el resultado de este experimento.

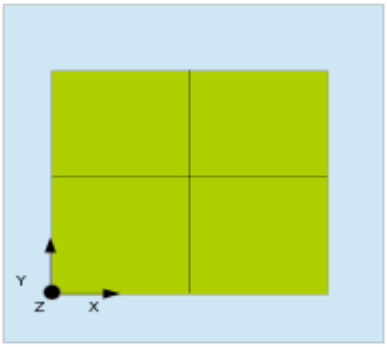


Figura 4.25: Estado de la pecera en esta primera parte del experimento: Vacia.

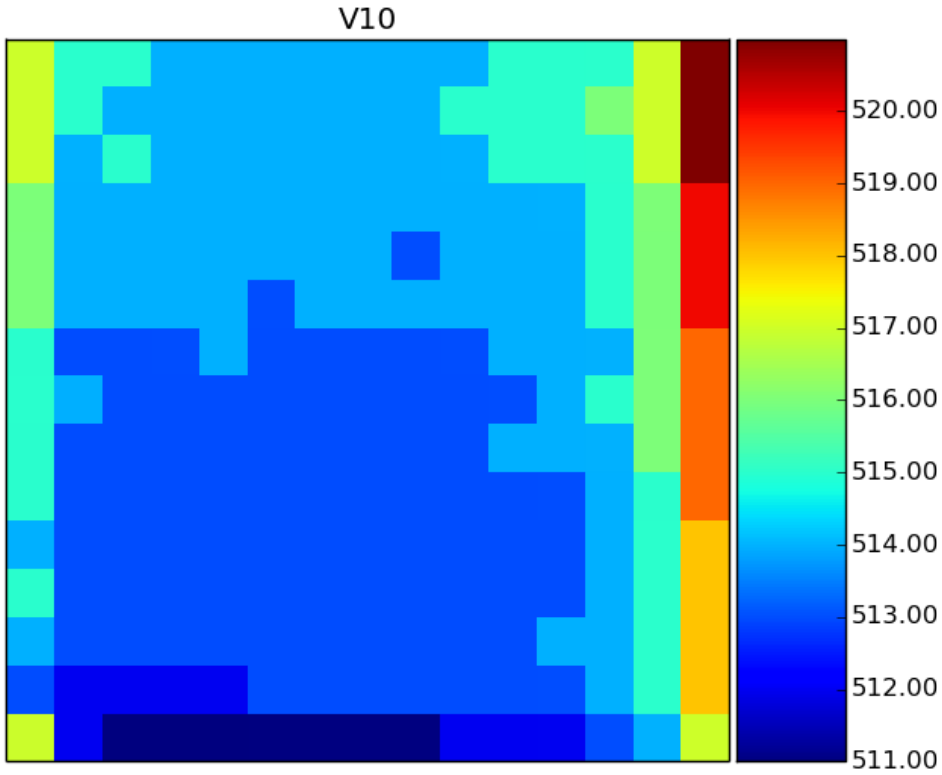


Figura 4.26: Barrido de 15x15 en el plano XY de V10 con detección de objetos pero sin objetos en el acuario. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

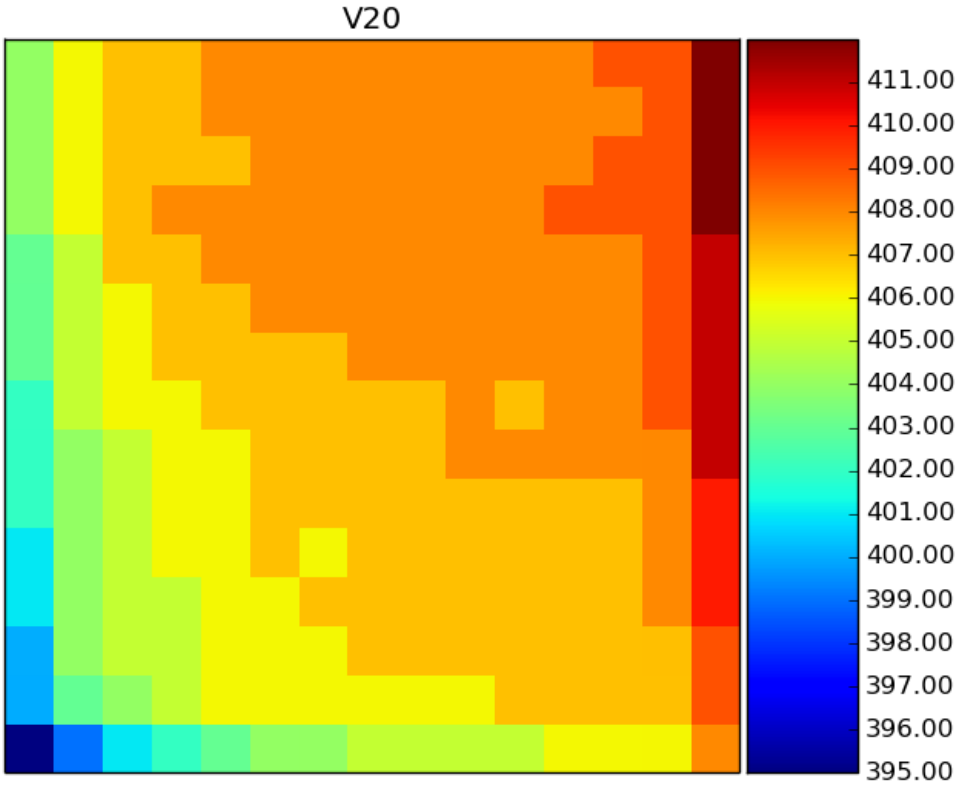


Figura 4.27: Barrido de 15x15 en el plano XY de V20 con detección de objetos pero sin objetos en el acuario. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

En este caso, de la misma manera que en la otra prueba realizada sin objetos, los resultados son los mismos. En este caso, sin embargo, se comprueba que el sistema no da falsos positivos.

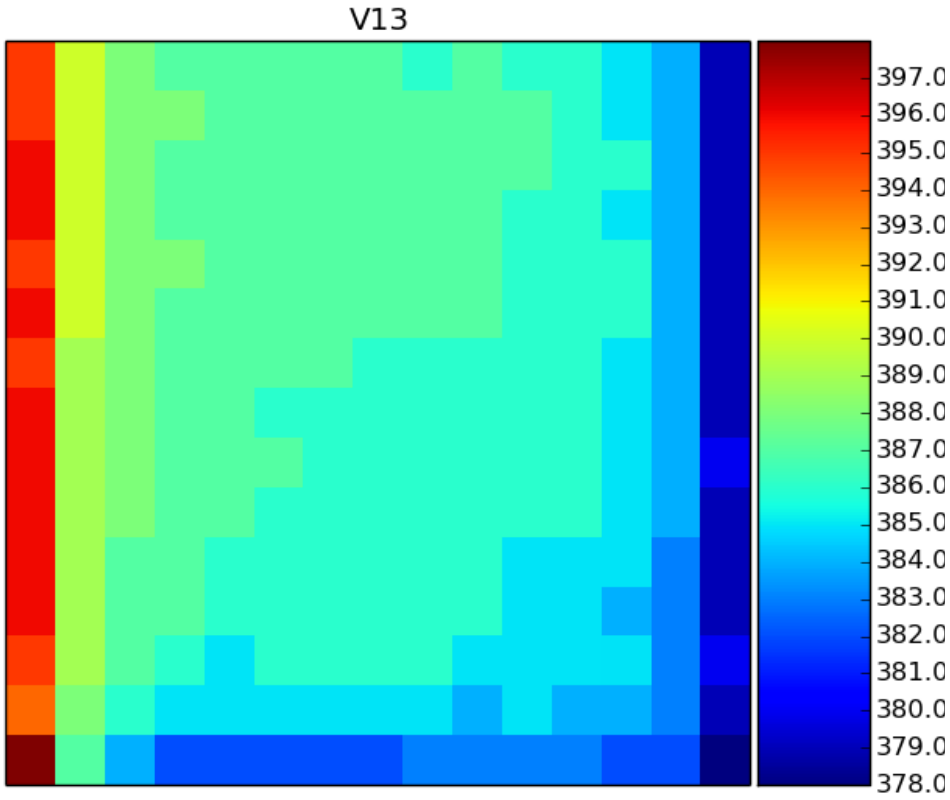


Figura 4.28: Barrido de 15x15 en el plano XY de V13 con detección de objetos pero sin objetos en el acuario. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

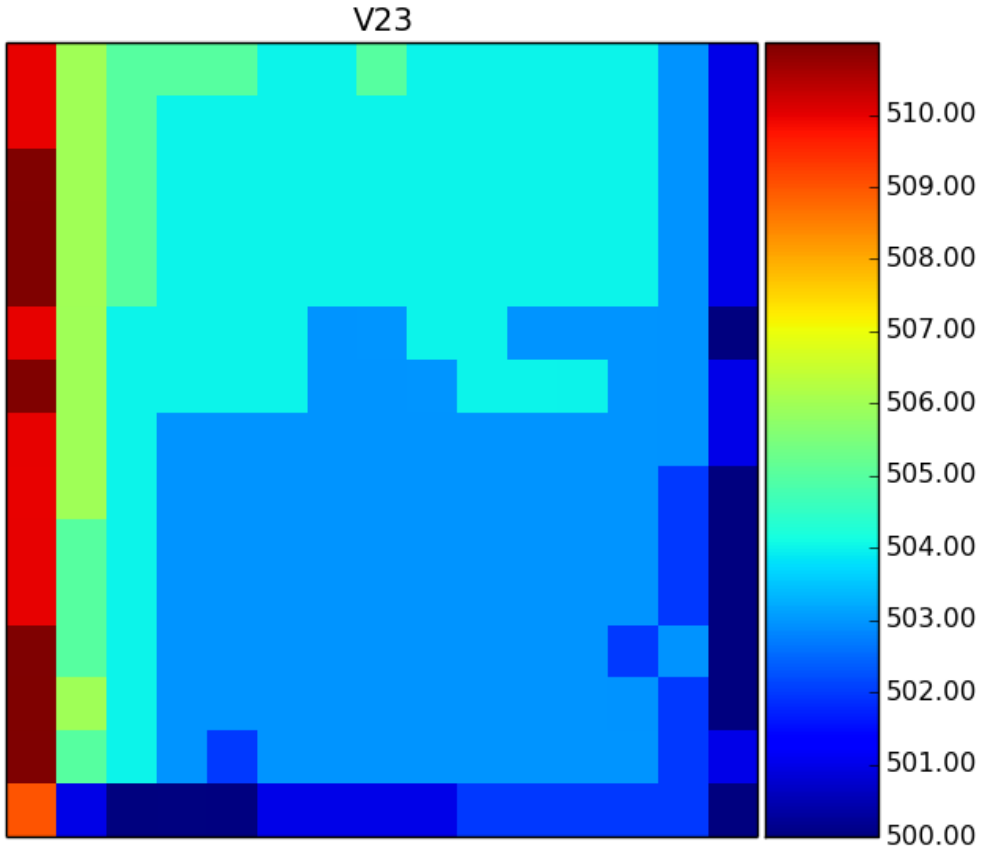


Figura 4.29: Barrido de 15x15 en el plano XY de V23 con detección de objetos pero sin objetos en el acuario. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

4.4.2 Detectando un bote de vidrio al fondo (45º) de la pecera

En este caso, se realiza la misma prueba situando un bote aislante en la misma altura a la que se realiza el barrido, girado y al fondo de la pecera (Figura 4.30). Se puede observar a continuación, en las Figuras 4.31 y 4.32 el resultado de este experimento.

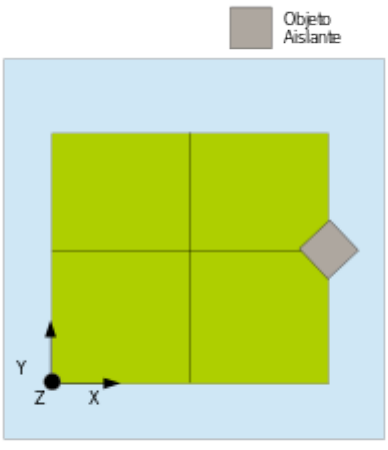


Figura 4.30: Estado de la pecera en esta primera parte del experimento: Se ha situado el objeto aislante al fondo de la pecera y con un ángulo de giro de 45°

Tal y como se muestra, el sistema detecta el objeto posicionado en ese lugar del acuario, deteniéndose en el momento de la detección.

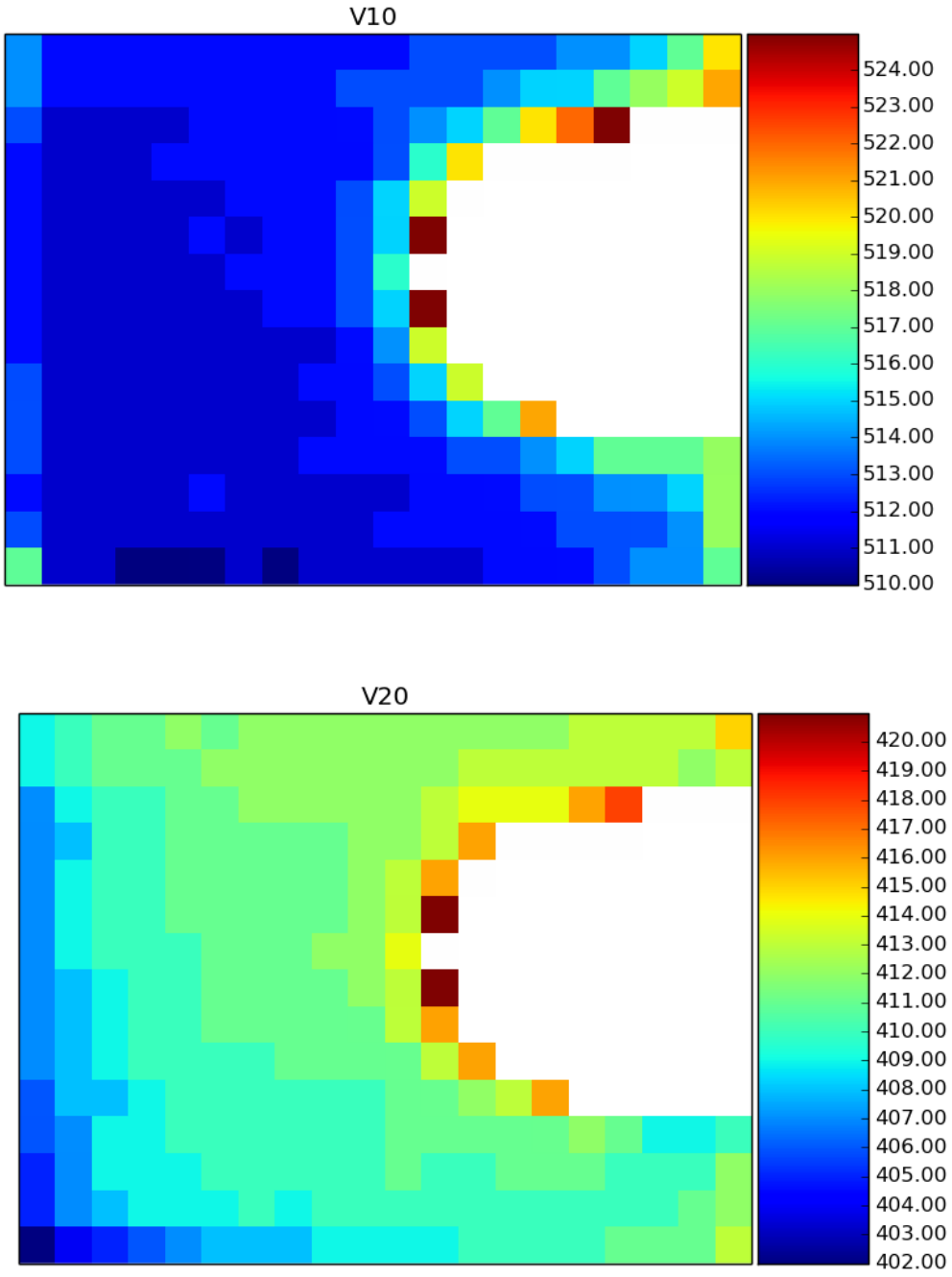


Figura 4.31: Barrido de 20x15 en el plano XY de las medidas V10 y V20 con detección de objetos y un obstáculo aislante situado en un lateral de la pecera. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

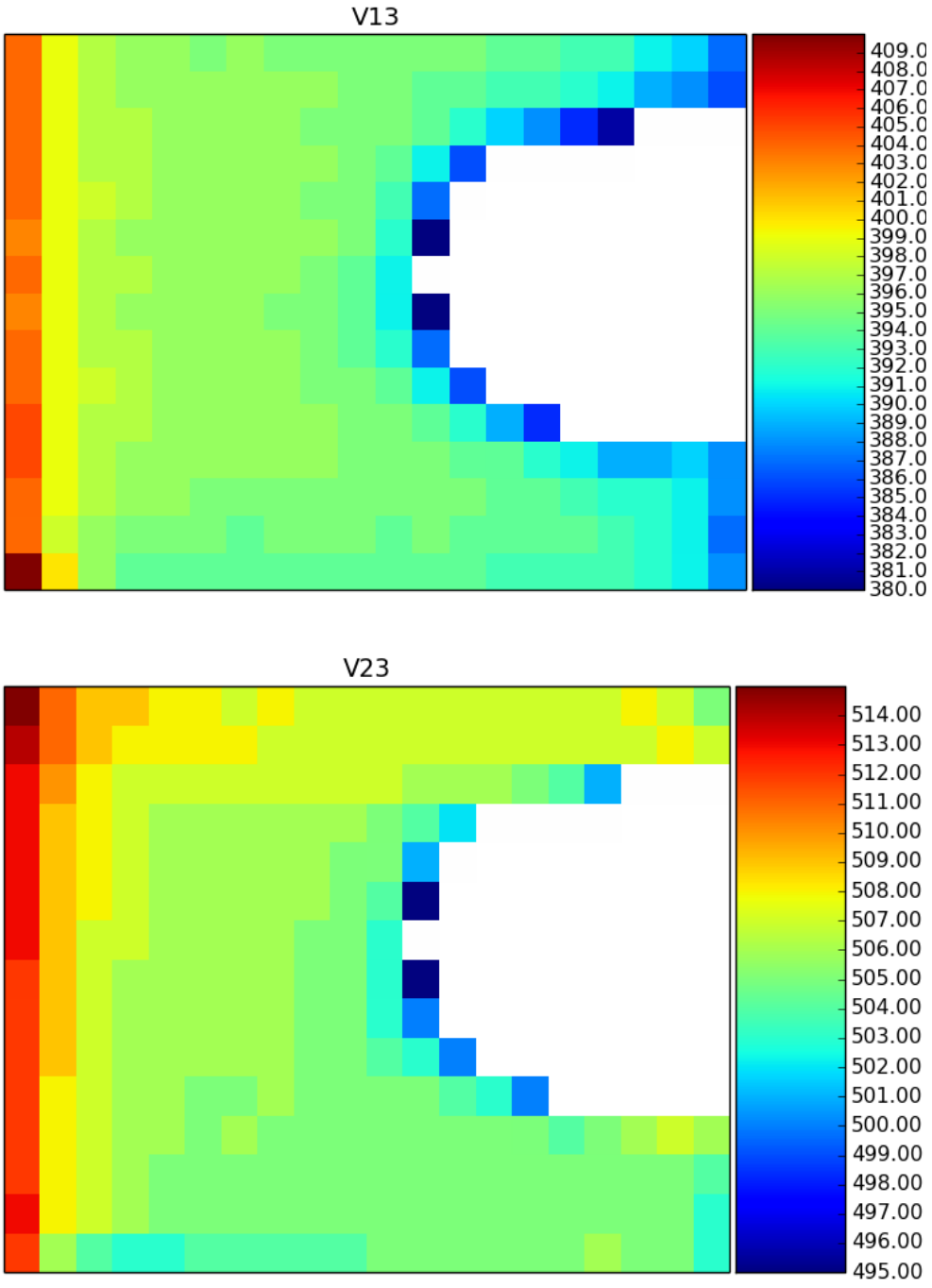


Figura 4.32: Barrido de 20x15 en el plano XY de las medidas V13 y V23 con detección de objetos y un obstáculo aislante situado en un lateral de la pecera. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

4.4.3 Detectando un bote de vidrio cuadrado a 0º en una esquina

En este caso, se realiza la misma prueba de detección situando un bote aislante en la misma altura a la que se realiza el barrido, esta vez perpendicular al sensor y en una esquina (Figura 4.33). Se puede observar a continuación, en las Figuras 4.34 y 4.35 el resultado de este experimento.

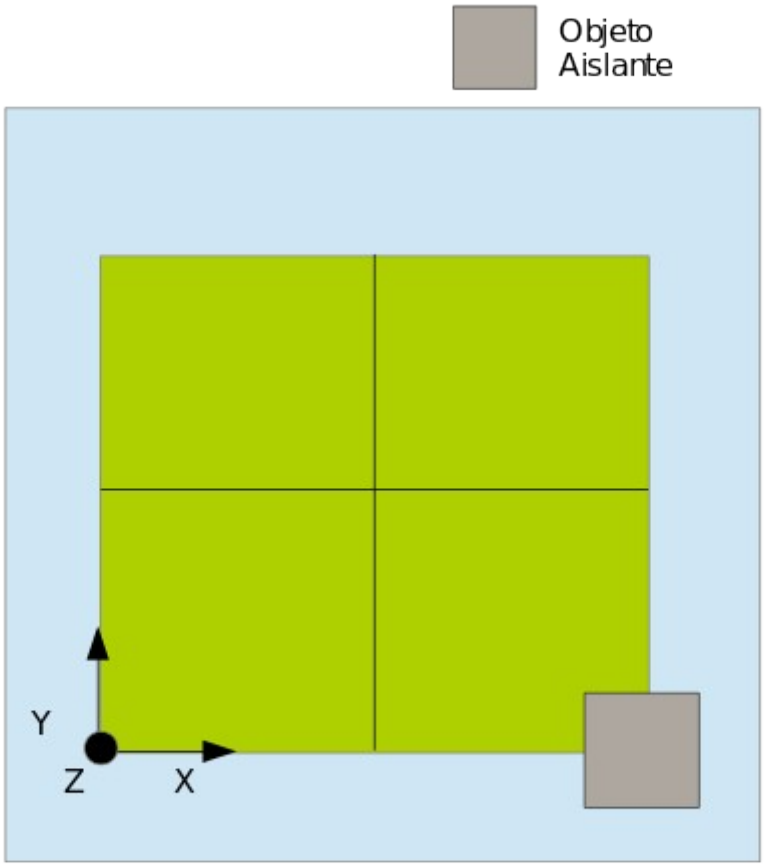


Figura 4.33: Estado de la pecera en esta fase del experimento: Se sitúa un objeto aislante en una esquina de la pecera

En este caso, los resultados son como se esperan, existiendo una detección positiva del objeto en la posición donde ha sido colocado. Es importante notar que las dimensiones del sensor juegan un papel importante, ya que en los lugares donde se produce la detección, son aquellos donde la parte frontal detecta el objeto, estando de cara al sistema el sensor en la posición central. Por este motivo, los objetos detectados se visualizarán como regiones de un tamaño mayor al real. Este efecto es fácilmente resoluble incluyendo este efecto en el algoritmo de detección y posicionamiento.

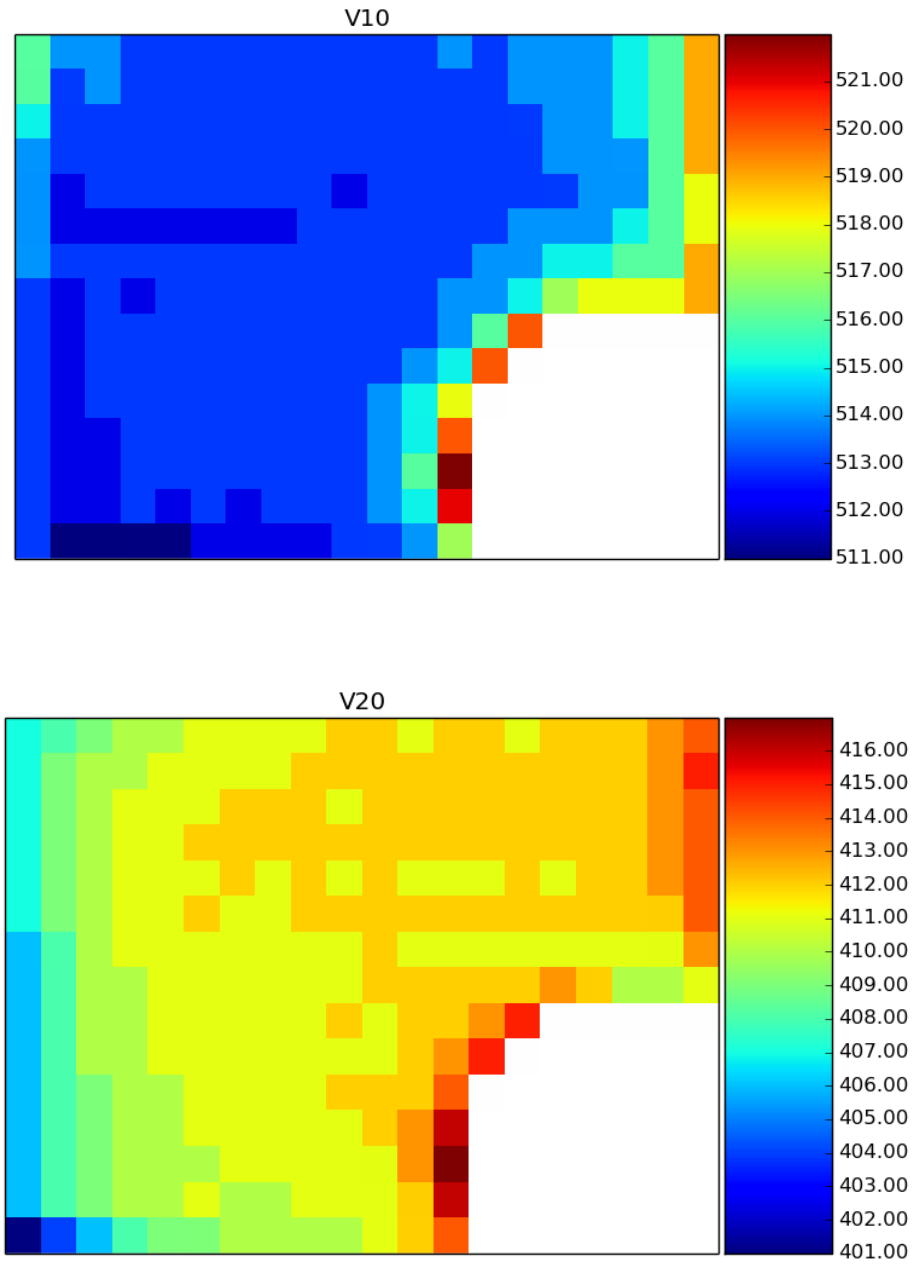


Figura 4.34: Barrido de 20x15 en el plano XY de las medidas V10 y V20 con detección de objetos y un obstáculo aislante situado en una esquina de la pecera. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

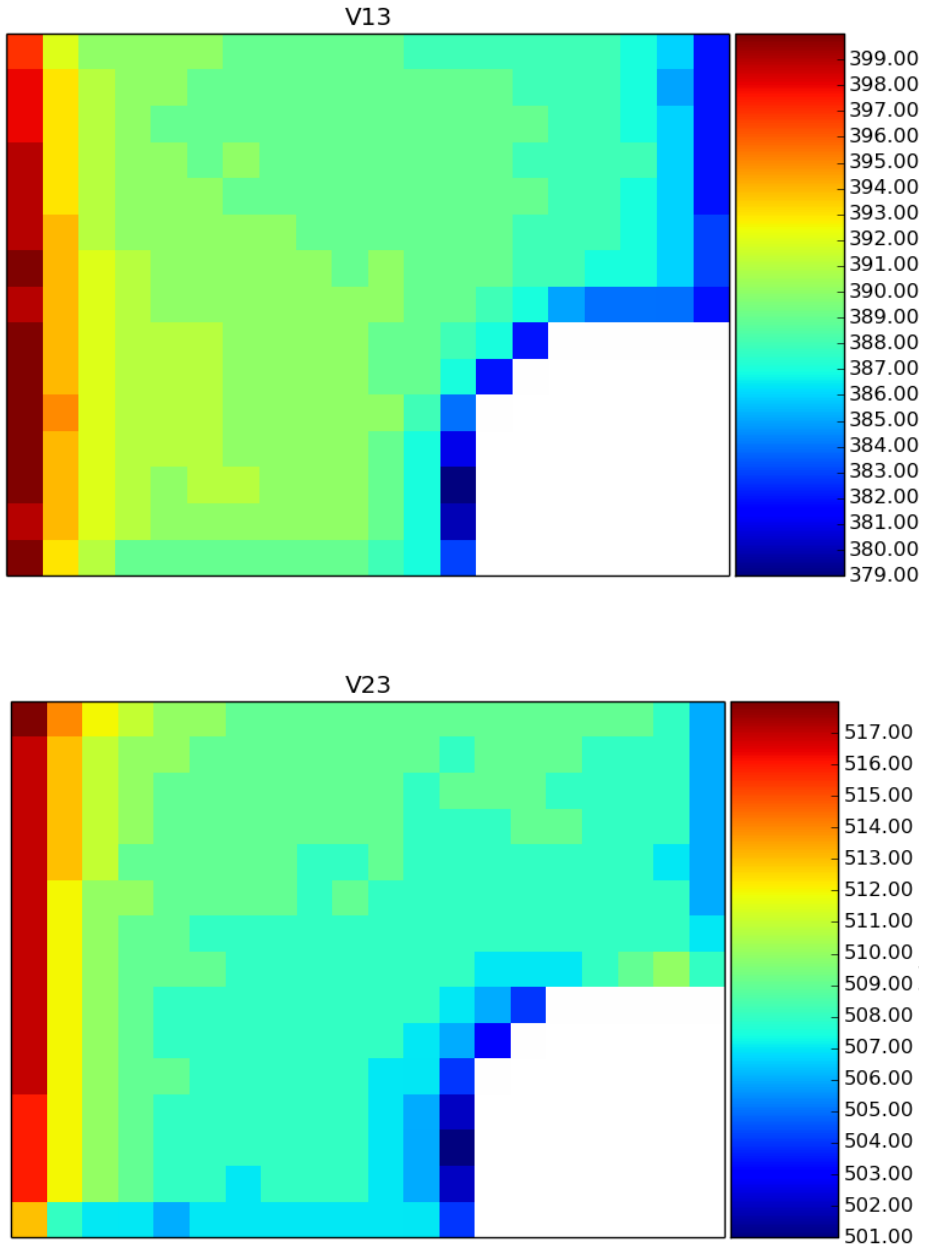


Figura 4.35: Barrido de 20x15 en el plano XY de las medidas V13 y V23 con detección de objetos y un obstáculo aislante situado en una esquina de la pecera. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

4.4.4 Detectando un bote de vidrio en una esquina de la pecera y un elemento extraño en la otra.

En este caso (Figura 4.36), se realiza la misma prueba que en el caso anterior pero situando además un filtro de agua de pecera a la misma altura a la que se realiza el barrido. Se puede observar a continuación en las Figuras 4.38 y 4.37 el resultado de este experimento.

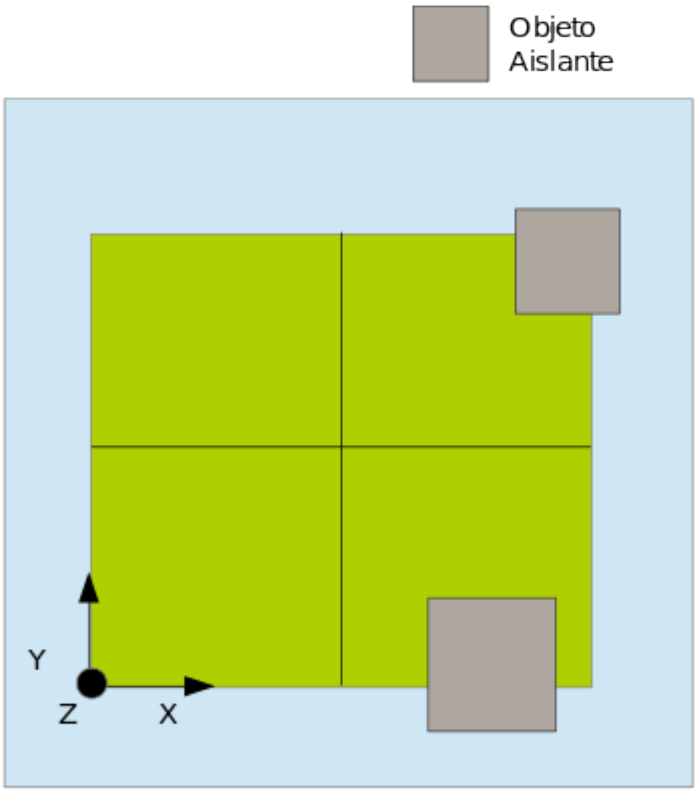


Figura 4.36: Estado de la pecera en esta fase del experimento: Se sitúa un objeto aislante en una esquina de la pecera

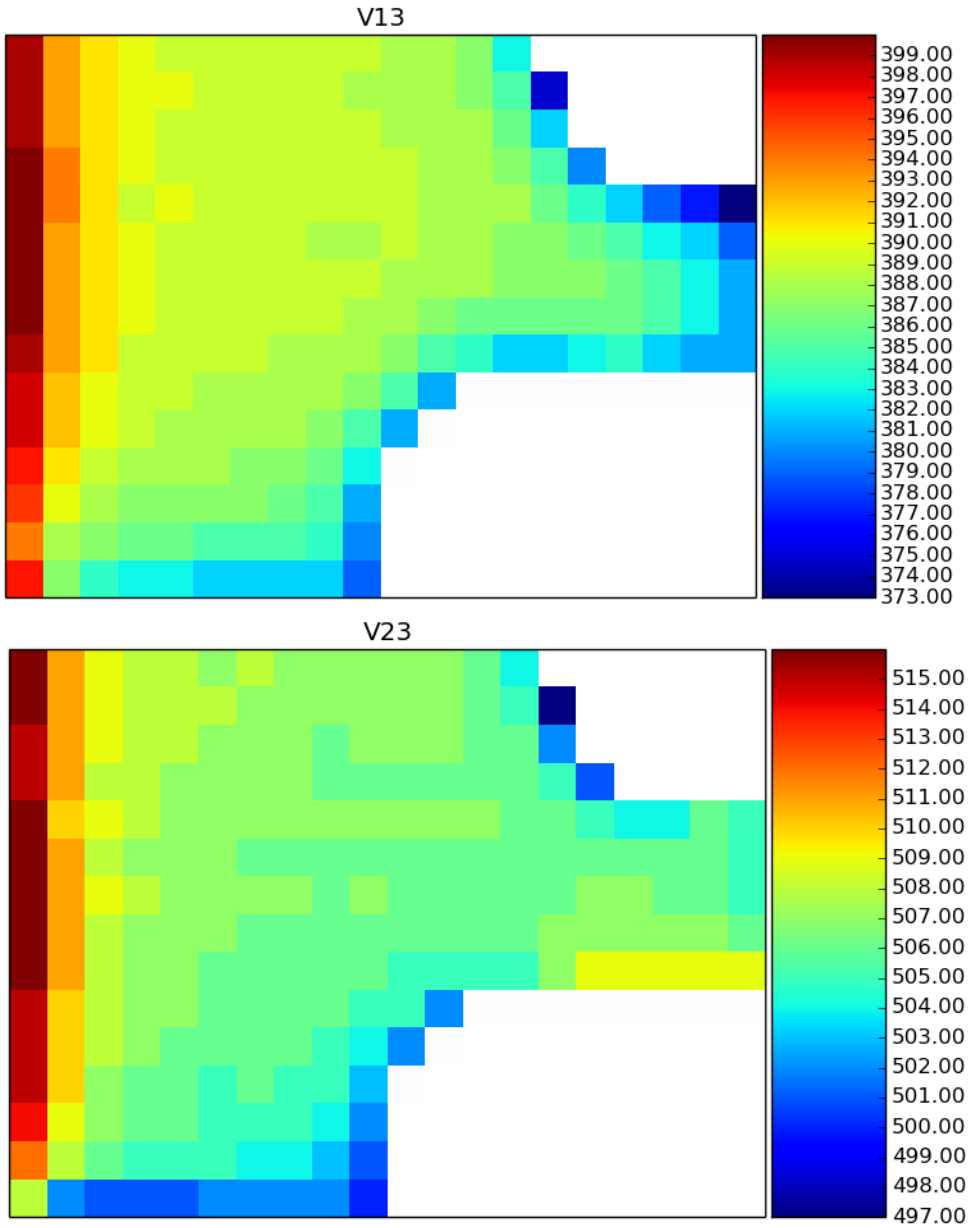


Figura 4.37: Barrido de 20x15 en el plano XY de las medidas V13 y V23 con detección de objetos y dos obstáculos en las esquinas de la pecera. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

En este experimento los resultados son tal y como se esperan, habiéndose detectado sin mayor problema el elemento nuevo introducido en el área de navegación. Se observa el efecto de zona ciega detrás del elemento inferior, que tiene unas dimensiones bastante menores a las que se intuyen en el gráfico mostrado.

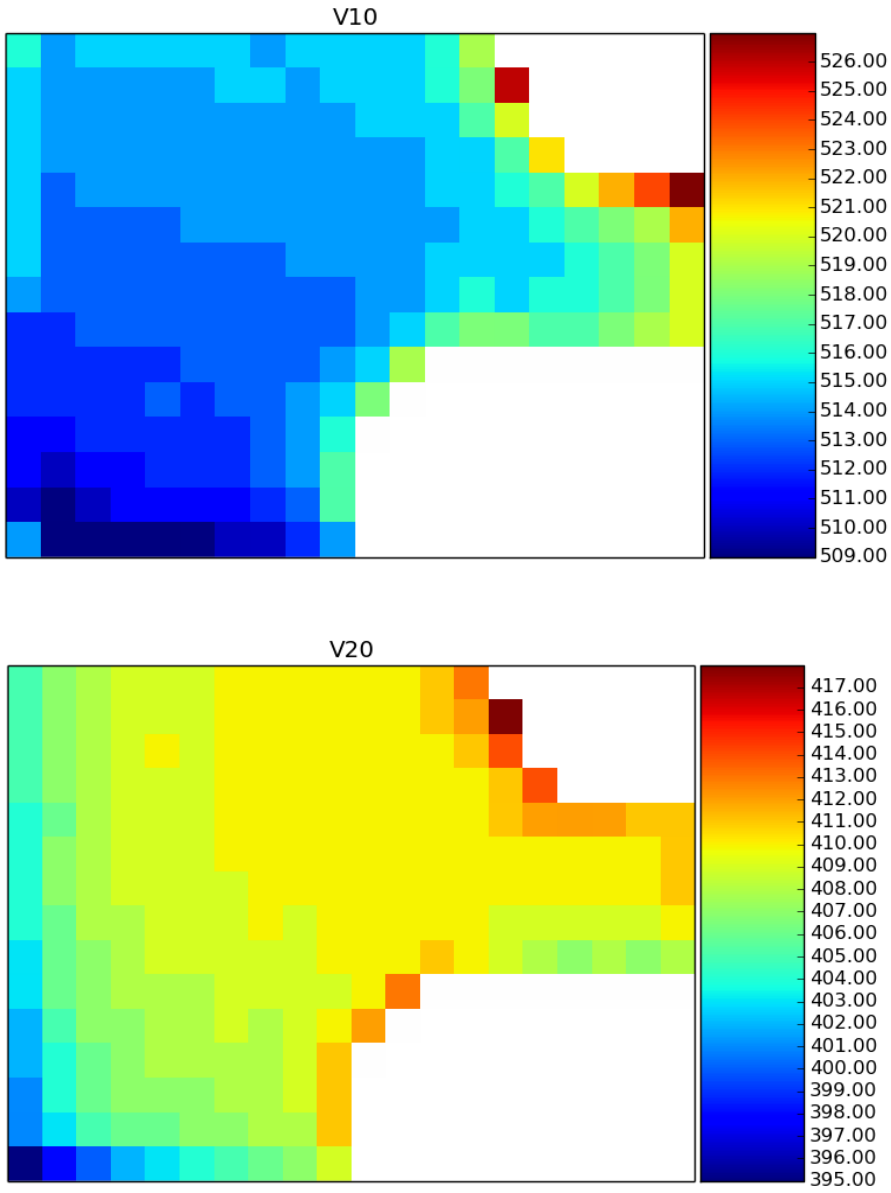


Figura 4.38: Barrido de 20x15 en el plano XY de las medidas V10 y V20 con detección de objetos y dos obstáculos en las esquinas de la pecera. Los valores mostrados de voltaje representan el valor medido por el conversor analógico de la placa arduino, que trabaja con 1024 valores para el rango de 0 a 5v.

4.4.5 Conclusiones extraídas del experimento

Tras este experimento, se concluye que el sistema:

- Es capaz de trabajar con mallados en 2D incompletos debido a la presencia de objetos
- No genera falsos positivos ante situaciones no contempladas
- Es capaz de detectar objetos en sus inmediaciones.

Experimentos

- Es capaz de reaccionar ante la presencia de objetos
- Puede sin mayor problema detectar objetos puestos arbitrariamente por la pecera
- Debido al algoritmo de mallado diseñado, genera una sombra detrás del objeto sin explorar. Esto se puede corregir sin mayor complicación cambiando el algoritmo de búsqueda implementado.
- De la misma forma, las dimensiones del sensor no se tienen en cuenta en el algoritmo y esto produce que las dimensiones resultado del objeto sean más grandes de las reales. Esto se puede corregir también teniendo en cuenta la posición real de los electrodos con respecto al centro de coordenadas del sensor.

Una carpeta con vídeos correspondientes a algunos de los experimentos realizados anteriormente podrá encontrarse en el repositorio online del proyecto, en una carpeta dedicada a ello (Ver anexo D).

5 Conclusiones generales y trabajo futuro

5.1 Conclusiones

A continuación se va a repasar la lista de objetivos para comentar a través de ellos cuales han sido los resultados finales de este trabajo:

- Objetivo Nº1: Diseño de un sistema mecánico capaz de soportar y desplazar por un acuario una sonda electro-receptora.

  Resultado: Se ha diseñado un sistema capaz de desplazarse en los ejes X, Y y Z y girar sobre sí mismo. Visto en la sección 3.2.

- Objetivo Nº2: Construcción de dicho sistema y adaptación al acuario.

  Resultado: Se ha construido AquaRide y se ha adaptado al acuario adquirido. Visto en la sección 3.2.3.

- Objetivo Nº3: Instalación en un ordenador del software y preparación hardware necesarios para el control de la posición del sistema mecánico en el acuario.

  Resultado: Se ha desarrollado el software necesario para el manejo de AquaRide y se ha adquirido una placa Sanguinololu para el control del mismo. Visto en las secciones 3.5 y 3.6.

- Objetivo Nº4: Diseño y construcción de una sonda electro-receptora que permita ser modificable y personalizable para distintos modelos de electro-localización.

  Resultado: Se ha diseñado un sensor paramétrico y se ha construido con éxito. Visto en la sección 3.4.

- Objetivo Nº5: Preparación de los elementos que forman la cadena de adquisición y procesado de datos

  Resultado: Se ha conectado el sistema de adquisición y creado el código en Python necesario para procesar los datos recibidos. Visto en la sección 3.6.

- Objetivo Nº6: Validación del sistema completo mediante la realización de experimentos
  - Calibración y validación del sistema de desplazamiento CNC

  Resultado: Tras varias correcciones en el diseño, el resultado final permite colocar el sensor con precisión de milímetros en cualquier lugar de la pecera. Visto en la sección 3.

  - Validación de la sonda eléctrica

  Resultado: Los experimentos han validado el modelo eléctrico que se ha utilizado para el sensor, abriendo las puertas a nuevos experimentos y tipos de sensor. Visto en la sección 4.

  - Validación de la fase de procesamiento de los datos obtenidos

Resultado: Una vez obtenidos los datos, se ha podido representar con éxito los resultados de distintos experimentos. Sección 4:

- Objetivo Nº7: Modificar el movimiento de la sonda en base a la información recogida por la misma.

  Resultado: Se ha implementado un pequeño algoritmo que realiza detección de objetos aislantes en la trayectoria e interrumpe su movimiento, permitiendo así realizar una exploración segura de la pecera. Sección 4.4:

En resumen, se ha construido un sistema de exploración subacuático basado exclusivamente en el sentido eléctrico utilizando un sensor artificial paramétrico soportado por una plataforma también paramétrica y replicable gracias a la tecnología de impresión 3D.

El resultado del trabajo ha sido positivo, ya que ha sido posible detectar con éxito objetos en las inmediaciones del sensor, así como las variaciones producidas por éstos.

Se han desplegado por tanto gracias a este proyecto una serie de vías de investigación en cuanto a métodos de exploración subacuáticos, modelos eléctricos del sensor más complejos y una infinidad de experimentos distintos que nos ayuden a comprender los métodos más efectivos y precisos de electro-localización, culminando en la realización de un sistema de exploración de movimiento autónomo en el que se combinen las técnicas más conocidas como el sónar o las cámaras de vídeo con la exploración eléctrica del medio.

5.2 Líneas de trabajo abiertas en este proyecto

Ante los resultados de este Trabajo de Fin de Máster se abren una infinidad de distintas direcciones en las que encaminar esta investigación. A continuación se detallan las opciones que parecen más interesantes de cara al estudio de la electro-localización.

5.2.1 Emisión de pulsos en movimiento.

Una de las mejoras más sencillas que se pueden realizar en este sistema es el cambio en los algoritmos implementados en la placa Sanguinololu para permitir la realización de medidas al mismo tiempo que se está moviendo el sensor. Esto permite un sensado continuo del medio y la modificación de la interacción con el medio de una manera mucho más natural, útil y similar a la realizada por los peces eléctricos.

5.2.2 Modificación del tipo de pulsos emitidos en función del medio.

Como continuación a la primera mejora enunciada, una línea interesante de desarrollo sería el estudio de la efectividad y beneficios de modificar la forma, la amplitud, la frecuencia o la fase de los pulsos emitidos en función de la presencia de objetos o seres vivos.

5.2.3 Desarrollo de un modelo eléctrico más sofisticado de la sonda electro-receptora para mejorar la imagen eléctrica del medio formada

Otra linea distinta de avance en este proyecto sería la mejora del modelo eléctrico utilizado y el desarrollo de una sonda más compleja que permita un escaneado del medio más preciso. Esto permitiría, por ejemplo, la creación de un algoritmo que permita la detección de múltiples objetos.

5.2.4 Creación de distintas sondas electro-receptoras con funciones distintas

Siguiendo con la idea de modificar el sensor utilizado, surgen varios tipos de sensores cuyo análisis puede ser interesante:

- Agrupamiento de sensores para detección de fondos o superficies
- Modelos bioinspirados más parecidos a una determinada especie de pez eléctrico
- Generalización del modelo eléctrico de la sonda para permitir otro tipo de sensados más complejos, como un sensor con cien electrodos.

5.2.5 Interacción con peces eléctricos vivos

Alternativamente a todas las opciones anteriores, existe la posibilidad de utilizar el sistema para interactuar directamente con los peces elefante u otro tipo de peces eléctricos.

5.2.6 Uso de la conductividad del agua para modificar la interacción con un ambiente no estructurado

Otra mejora, en este caso más técnica, es el uso del dato de la conductividad del agua para obtener un valor de referencia que permita al sensor detectar si la exploración del medio, por ejemplo, ha empezado en un lugar donde ya hay obstáculos presentes.

5.2.7 Modificación de la amplitud de los pulsos emitidos para variar la resolución y el rango de detección

Como última línea de estudio propuesta, se propone utilizar una de las propiedades de las que no se ha hablado específicamente en este proyecto, que es la variación del rango de alcance del pez en función de la diferencia de potencial aplicado. Esto puede permitir mayores alcances en zonas despejadas y alternativamente mayores precisiones en zonas de posicionamiento crítico.

6 Referencias

[1] B.Kramer, “Electro-communication in teleost fishes”, 1990.

[2] Theodore H. Bullock,Carl D. Hopkins,Arthur N. Popper,Richard R. Fay (eds.), “Electroreception”, 2005.

[3] Marielle Thomas, André Florion, Didier Chrétien, Denis Terver, “Real-time biomonitoring of water contamination by cyanide based on analysis of the continuous electric signal emitted by a tropical fish: Apteronotus albifron”, Water research Magazine, Volumen 30, Edición 12, Páginas 3083–3091, Diciembre 1996.

[4] Frédéric Boyer, Pol Bernard Gossiaux, Brahim Jawad, Vincent Lebastard, and Mathieu Porez , “Model for a sensor Inspired by Electric”, IEE Transactions on robotics, Vol. 28, Num. 2, Abril 2012.

[5] Dominic Clarke, Heather Whitney, Gregory Sutton, Daniel Robert, “Detection and Learning of Floral Electric Fields by Bumblebees”, Science Magazine, Vol 340, Num. 6128, pags 66-69, 5 Abril, 2013.

[6] A A Caputi, R Budelli, K Grant and C C Bell, “The electric image in weakly electric fish: physical images of resistive objects in Gnathonemus petersii”, The Journal of Experimental Biology, Volumen 201, pags 2115–2128, 1998.

[7] Stephen M. Kajiura and Kim N. Holland, “Electroreception in juvenile scalloped hammerhead and sandbar sharks ”, The Journal of Experimental Biology volumen 205,paginas 3609–3621, 2002.

[8] Bernd Kramer, “Electroreception and communication in fishes”, Revista Progress in Zoology, volumen 42, 1996.

[9] Ana Carolina Pereira, Angel Ariel Caputi, “Imaging in Electrosensory Systems”, Interdiscip Sci Comput Life Sci, Vol 2, pags 291-307, 2010.

[10] Carl D. Hopkins , “Neuroethology of electric communication ”, Ann. Rev. Neurosci., pags. 497-535 Noviembre 1988.

[11] Gerhard Von Der Emde, “Active electrolocation of objects in weakly electric fish”, The Journal of Experimental Biology, Vol. 202, pags. 1205-1215, 1999.

[12] Caroline García Forlim, Lirio O B Almeida, Pablo Varona, Francisco De Borja Rodríguez, Reynaldo Daniel Pinto, “Study of electric and motor behaviour in weakly electric fish, Gymnotus carapo and Gnathonemus Petersii, using Information Theory”, 2012.

[13] Gerhard von der Emde, “Discrimination of objects through electrolocation in the weakly electric fish, Gnathonemus Petersii”, Journal of comparative physiology, Vol 167, pags. 413-421, 1990.

[14] Stephan Schwarz,Gerhard von der Emde , “Distance discrimination during active electrolocation in the weakly electric fish Gnathonemus petersii”, Journal of comparative physiology, Vol 186, pags. 1185-1197 , 1990.

[15] Kramer, B., 1979, “Electric and Motor responses of the Weakly electric fish, Gnathonemus petersii (Mormyridae), to play-back of social signals”, Behav. Ecol. Sociobiol., 6(1), pags. 67– 79.

[16] Noël Servagent, “First results on a sensor bio-inspired by electric fish”, 2012.

[17] Yonatan Silverman, James Snyder, Yang Bai and Malcolm A. MacIver “Location and Orientation Estimation with an Electrosense Robot”, 2012

[18] Yang Bai , James Snyder, Yonatan Silverman, Michael Peshkin and Malcolm A. MacIver, “Sensing Capacitance of Underwater Objects in Bio-inspired Electrosense”, 2012

[19] Izaak D. Neveln, Yang Bai, James B. Snyder, James R. Solberg, Oscar M. Curet, Kevin M. Lynch and Malcolm A. MacIver, "Biomimetic and bio-inspired robotics in electric fish research ", 2013, The Journal of Experimental Biology vol 216 (13). pags 2503-2514

[20] Steven Bruce Ferrer, Vincent Lebastard and Frédéric Boyer, "Autonomous Underwater Docking Using Active/Passive Electro-location ", 2016

[21] V. Lebastard, C. Chevallereau, A. Girin, F. Boyer and P.B. Gossiaux, "Localization of small objects with electric sense based on kalman filter ", 2012

[22] Vincent Lebastard, Christine Chevallereau, Alexis Girin, Noël Servagent, Pol-Bernard Gossiaux and Frédéric Boyer , "Environment reconstruction and navigation with electric sense based on a Kalman filter ", The International Journal of Robotics Research 2013 32: 172

[23] James R. Solberg , Kevin M. Lynch , Malcolm A. MacIver , "Active Electrolocation for Underwater Target Localization", The International Journal of Robotics Research Vol. 27, No. 5, May 2008, pp. 529–548

[24] Yannick Morel, Vincent Lebastard, and Frédéric Boyer, "Neural-based Underwater Surface Localization through Electrolocation ", International Conference on Robotics and Automation (ICRA) , Stockholm, Sweden, May 16-21, 2016

[25] Página oficial de CoreXY http://corexy.com/ . Último acceso 20/06/2016

[26] Referencia de Sanguinololu en la wiki de RepRap: URL:http://reprap.org/wiki/Sanguinololu . Último acceso 20/06/2016

[27] Página oficial de Openscad http://www.openscad.org/about.html. Último acceso 20/06/2016

[28] Zupanc, G.K.H., Sîrbulescu, R.F., Nichols, A., Ilies, I.V.: Electric interactions through chirping behavior in the weakly electric fish, *Apteronotus leptorhynchus*. *Journal of Comparative Physiology A* **192**, 159-173 (2006)

[29] James Y. Valentino, Joseph Goldenberg "Introduction to computer numerical control (CNC)", 2003

[30] Página oficial del Natural Environment Research Council. URL: http://www.nerc.ac.uk. . Último acceso 20/06/2016

[31] Michael S. Lewicki, Bruno A. Olshausen, Annemarie Surlykke and Cynthia F. MossMichael S. Lewicki, Bruno A. Olshausen, Annemarie Surlykke and Cynthia F. Moss, "Scene analysis in the natural environment", 2014

[32] Hideo Kodama, "Automatic method for fabricating a three-dimensional plastic model with photo-hardening polymer," *Review of Scientific Instruments*, Vol. 52, No. 11, pp. 1770–1773, November 1981

[33] Peter Moller, "Multimodal sensory integration in weakly electric fish: a behavioral account", 2002

[34] Forlim, C. G., & Pinto, R. D. (2014). Automatic realistic real time stimulation/recording in weakly electric fish: long time behavior characterization in freely swimming fish and stimuli discrimination. *PloS one*, *9*(1), e84885.

[35] Forlim, C. G., Pinto, R. D., Varona, P., & Rodríguez, F. B. (2015). Delay-Dependent Response in Weakly Electric Fish under Closed-Loop Pulse Stimulation. *PloS one*, *10*(10), e0141007.

[36] A. Lareo, P. Varona, F.B. Rodriguez. Weakly Electric fish information processing analyzed through closed-loop code-driven stimulation. AIMS 2014. Special Session 77: Theoretical, Technical and Experimental Challenges in Closed-Loop Approaches in Biology, 1. Madrid, Spain.

[37] Lareo Fernández, Á. (2014). Estudio del procesamiento de información en peces eléctricos mediante técnicas lazo cerrado de estimulación en tiempo real. Trabajo Fin de Máster (http://hdl.handle.net/10486/663284).

[38] C.G. Forlim, C. Muñiz, R.D. Pinto, F.B. Rodríguez, P. Varona. 2013. Behavioral driving through on line monitoring and activity-dependent stimulation in weakly electric fish. *BMC Neuroscience* 2013, 14: P405 (CNS 2013, Paris).

[39] C. Muniz, C.G. Forlim, R.T. Guariento, R.D. Pinto, F.B. Rodriguez and Pablo Varona. 2011. Online video tracking for activity-dependent stimulation in neuroethology. BMC Neuroscience, 2011, 12:P358 (CNS 2011, Stockholm).

[40] P. Chamorro, C. Muniz, R. Levi, D. Arroyo. F.B. Rodriguez, P. Varona. 2012. Generalization of the dynamic clamp concept in neurophysiology and behavior. *PLoS ONE* 7(7): e40887.

A. Presupuesto

Diseño, construcción y montaje del sistema de desplazamiento en el acuario

Repuesto de plástico PLA para la impresora 3D.....50€
Placa Sanguinololu.....40€
Drivers POLOLU A4988(x3).....18€
Motores Stepper (x3).....30€
Servomotores (x2).....16€

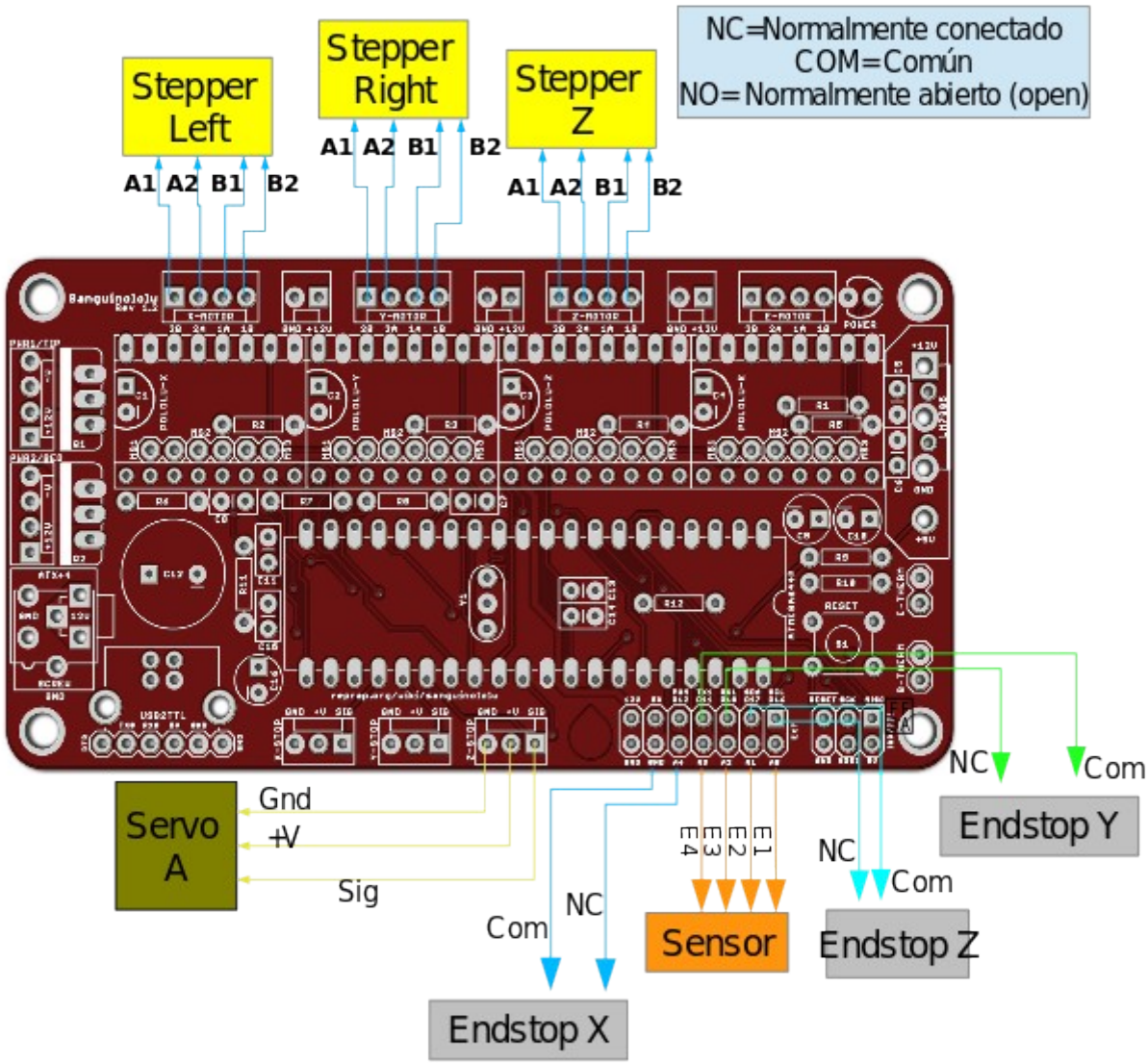
Pecera cúbica 45cm.....45€
Hilos de Nylon.....4€
Detectores de fin de carrera.....4€
Varillas de metal.....1,50€
Organizador de cables.....1€
Arandelas, tornillos y tuercas de A. Inoxidable.....15€
Cable, conectores, y contactos de crimpado.....10€
Cinta de teflón para suavizado de poleas.....1€
Pegamento Loctite SuperGlue3.....5€

Material de oficina.....20€
Licencia para OpenSCAD.....0€

Total Presupuesto.....260,50 €

B. Conexiones y montaje

I. Conexiones a la placa Sanguinololu



Aclaración sobre los endstops

Se ha de colocar la salida asociada a tierra de Sanguinololu (“endStopYGND”, por ejemplo) al pin del endstop “COM” y el pin asociado a señal (“endstopYSIG”, por ejemplo) al pin de Normalmente Conectado. De esta manera, cuando el fin de carrera se active, el circuito se abre y se encuentra un 1 en el sistema, que se interpreta como tal. Esto es una medida de seguridad

para evitar que si un cable de detector de fin de carrera se suelta o se rompe, AquaRide no siga avanzando indefinidamente, sino que se pare.

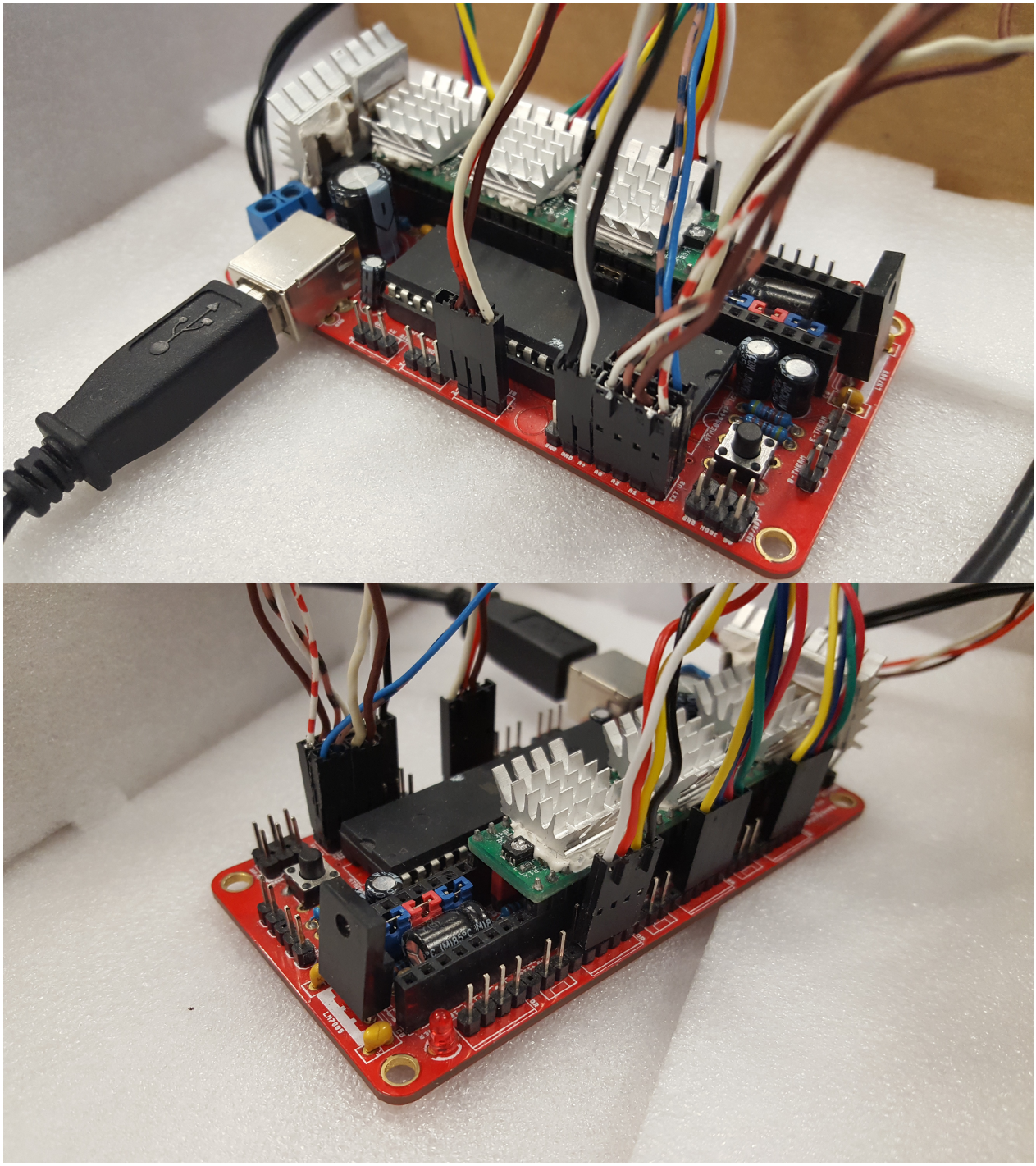


Figura B.1: Fotografías de la placa Sanguinololu con las conexiones hechas

II. Montaje de las piezas 3D

Para realizar el ensamblado de las piezas, es necesario seguir este orden para evitar bloqueos.

1. Encajar las piezas “Top_corner” y “Top_corner_Motor” a la pecera.
2. Atornillar los motores paso a paso a las piezas “Top_Corner_Motor”.
3. Poner las piezas “Motor_Wheel” en los ejes de los motores atornillados.
4. Introducir las barras de acero en una de las dos piezas “Wagon”.
5. Introducir en las barras de acero la pieza “Carrier”.
6. Cerrar las barras de acero introduciéndolas en la otra pieza “Wagon”.
7. Poner el Servomotor del eje A en “Carrier”.
8. Poner la pieza “Eje_Transmisión_V2” en “Carrier.”.
9. Encajar el “Cilindro_Sujecion_Vertical” en “Eje_Transmisión_V2” por debajo de “Carrier” de tal manera que se permita un giro suave de “Eje_transmision_V2” sobre “Carrier”.
10. Pegar la “Rueda_Servo_A” en su servo. Tener en cuenta la posición actual del sistema ya que el servo usado no es de rotación continua y de esto depende su calibración.
11. Ensamblar las piezas de “Eje_Dentado”(asegurarse de que el conducto interior es transitable por los cables que luego se introducirán) e introducirlo por “Eje Transmisión_V2”. Esta pieza debe bajar y subir por su conducto con bastante suavidad, para evitar atranques durante el movimiento en Z.
12. Añadir la pieza “Endstop_Z” que sirve como tope en el eje Z superior.
13. En el caso de que se esté usando el mismo sensor empleado en este proyecto, unir las tuercas, arandelas y tornillos de acero inoxidable (Tamaño M8, tornillos de 16mm de largo), con las piezas correspondientes de “Probe” Entre las arandelas, conectar los cables que lleven la señal hasta la placa Sanguinololu. Estos cables deben ser bastante alargados, ya que tan sólo salir desde la pecera hasta fuera consume aproximadamente un metro de distancia. Es recomendable utilizar cables de distintos colores para asociar sin problemas los electrodos a los pines de entrada de la placa.
14. Introducir los cables de los electrodos por “Eje_Dentado” y dejarlos de momento al aire, ya que luego se tendrán que introducir en el organizador de cables.
15. Aprovechar para encajar el sensor en “Eje_Dentado”.

16. Colocar todo este sistema, soportado por las piezas “Wagon” en la pecera.
17. Colocar el motor paso a paso del eje Z en “Eje_Transmision_V2” (se recomienda usar un paso a paso menos alto, ya que este movimiento no requiere mucho par y es mejor cuanto menos pese, ya que desequilibra el centro de gravedad del eje.
18. Colocar los detectores de fin de carrera. (ver conexiones en el siguiente apartado).
19. Agrupar los cables del motor paso a paso del eje Z, el detector de fin de carrera Z y los cables del sensor .
20. Poner un organizador de cables en el hueco del “Top_Corner_Motor” y conectarlo al lugar destinado para ello en “Carrier”. Dejar suficiente longitud para que se pueda desplazar el CoreXY correctamente por toda la pecera.
21. Introducir todos los cables que se han colocado hasta ahora por el organizador de cables para sacarlos de la pecera hacia fuera.
22. Conectar los motores paso a paso y el servomotor a la placa Sanguinololu (Ver conexiones en siguiente apartado).
23. Probar conectando momentáneamente la placa Sanguinololu al PC que los motores funcionan correctamente (Alimentar la Sanguinololu con la fuente de alimentación, ya que sólo con la alimentación por USB no funcionan).
24. Una vez comprobado esto, desconectar la alimentación y cubrir las ruedas de los motores paso a paso del CoreXY con manguitos de goma, a ser posible. La idea es aumentar la fricción con el menor número de vueltas, ya que esto hace propicio que el cable de Nylon se enrede.
25. Atar los cables de Nylon (Cuanto más grueso mejor, el utilizado en este proyecto es de 1mm de diámetro) sobre el sistema. Para reducir la fricción, añadir cinta de teflón a todos los conductos por los que pasan los cables. Utilizar como referencia para atarlos la Figura 3.2.
26. Conectar los cables procedentes del acuario a la Sanguinololu (finales de carrera, motor Z y cables del sensor).
27. Ya está todo montado. Volver a alimentar la placa Sanguinololu. Utilizar los comandos por serie “Sample” para comprobar el funcionamiento de la parte de muestreo y las funciones “move” y “test” para comprobar el movimiento de AquaRide y su calibración.

c. Manual de puesta en funcionamiento del software

Una vez montado el circuito físico, es necesario poner en marcha el software necesario para controlar AquaRide. Para ello, se necesita:

- Instalar el IDE de Arduino y cargar el addon para la tarjeta Sanguino.
  - https://github.com/CarlosGS/grblForCyclone/tree/grblForCyclone/ArduinoIDE_addonForSanguinololu
- Instalar Python

En el caso de que queramos cargar el firmware en la placa, abrir el IDE de Arduino:

- Ir a la carpeta en la que se haya instalado Arduino y ejecutar "sudo ./arduino" (Si no se utiliza el comando sudo es posible que no se pueda cargar el firmware en la placa). Entonces, seleccionar cuidadosamente en el menú herramientas:
  - La placa "Sanguino". Si hemos instalado el addon correctamente, debería aparecer.
  - El puerto USB en el que está conectada la Sanguinololu
  - El procesador adecuado. En el caso de la placa Sanguinololu del laboratorio, corresponde a la opción "ATmega644P o Atmega644PA (16MHz)"

En el caso de que el firmware ya esté cargado en la placa, hay dos opciones para hacer funcionar AquaRide:

- Usar los scripts de Python que automáticamente envían los comandos necesarios por puerto serie (Asegurarse de que está bien seleccionado el puerto en el que se ha conectado por USB la Sanguinololu
- Usar el monitor serial del IDE de arduino para enviar los comandos manualmente. Este método es el mejor para tomar contacto en un primer momento y entender la interfaz de AquaRide. En caso de duda enviar el comando "-help".

D. Código actualizado

Se pueden descargar o consultar las versiones actualizadas del código usado para para AquaRide en "https://github.com/hasshido/AquaRide". En este repositorio se encuentran tanto los scripts empleados de Python, como el firmware cargado en la placa Sanguinololu. También se puede encontrar en este repositorio los ficheros ".scad" que corresponden a las piezas desarrolladas en OpenSCAD y los ficheros de datos tipo ".csv" que corresponden a los resultados obtenidos de las pruebas de medida.

E. Código utilizado: Firmware para la placa Sanguinololu

Main.ino

```
#include <Servo.h>

// Position
long CoreXY_Pos[4] = {0, 0, 0, 90}; // X(steps), Y(steps, Z(steps), A(deg, 1-180)
const long Aquarium_stepsX = 15300;
const long Aquarium_stepsY = 13300;
const long Aquarium_stepsZ = 8300;

// Endstops connections
const int endStopXSIG = A4;
// const int endStopXGND = A1; //

const int Probe0 = A0; // Emitter 1 // Red and white
const int Probe1 = A1; // ADC 1     // brown
const int Probe2 = A2; // ADC 2     // brown and black
const int Probe3 = A3; // Emitter 2 // white

const int endStopYSIG = 10;
const int endStopYGND = 11;

const int endStopZSIG = 16; // PC0-SCL
const int endStopZGND = 17; // PC1-SDA

// Stepper connections
// In Driver Pins

const int enablePin = 14;
const int stepSpeed = 300; //usec
const int stepSpeedZ = 500; //more value means slower movement

// To set directions L=Left, R=Right, Z=Up
const int dirPinL = 21;
const int dirPinR = 23;
const int dirPinZ = 2;

// To step
const int stepPinL = 15;
const int stepPinR = 22;
const int stepPinZ = 3;

// Servo Connections
```

```
// In the Z-Stop Pins

const int servoPinA = 20;
const int ServoSpeedA = 10; //ms, more value means slower movement

Servo myservoA;

static inline int8_t sgn(int val) {
  if (val < 0) return -1;
  if (val == 0) return 0;
  return 1;
}

float AnalogReadAverage(int Pin, int Samples) {
  int i = 0;
  float Measures = 0;
  float Aux = 0;
  for (i = 0; i < Samples; i++) {
    Aux = analogRead(Pin);
    delay(1);
    Measures = Measures + Aux;
  }
  return Measures / Samples;
}

void GetVoltages(int Samples = 1) {
  long Vdip[5];  //Voltage in A1, A1, A2 with no voltage applied
  float V10; float V13;  //Voltage in A1, 5V in A0/A3
  float V20; float V23;  //Voltage in A2, 5V in A0/A3
  long i;
  long averageSamples = Samples;
  //for (i = 0; i < Samples; i++) {

    //A0 = 5v
    digitalWrite(Probe3, LOW);
    digitalWrite(Probe0, LOW);
    delay(1);
         //         Vdip[0]=AnalogReadAverage(Probe0,   averageSamples)-
AnalogReadAverage(Probe3, averageSamples);

    //A0 = 5v
    digitalWrite(Probe3, LOW);
    digitalWrite(Probe0, HIGH);
    delay(1);
    V10 = AnalogReadAverage(Probe1, averageSamples);
    V20 = AnalogReadAverage(Probe2, averageSamples);
         //         Vdip[1]=AnalogReadAverage(Probe0,   averageSamples)-
AnalogReadAverage(Probe3, averageSamples);

    digitalWrite(Probe3, LOW);
    digitalWrite(Probe0, LOW);
    delay(1);
         //         Vdip[2]=AnalogReadAverage(Probe0,   averageSamples)-
AnalogReadAverage(Probe3, averageSamples);
```

```
    //A1 = 5v
    digitalWrite(Probe3, HIGH);
    digitalWrite(Probe0, LOW);
    delay(1);
         //         Vdip[3]=AnalogReadAverage(Probe0,  averageSamples)-
AnalogReadAverage(Probe3, averageSamples);
    V13 = AnalogReadAverage(Probe1, averageSamples);
    V23 = AnalogReadAverage(Probe2, averageSamples);

    digitalWrite(Probe3, LOW);
    digitalWrite(Probe0, LOW);
         //         Vdip[4]=AnalogReadAverage(Probe0,  averageSamples)-
AnalogReadAverage(Probe3, averageSamples);

    // Send data via serial
    // Serial.print("Pos= ");
             Serial.print("DATA:\t");       Serial.print(CoreXY_Pos[0]);
Serial.print("\t"); Serial.print(CoreXY_Pos[1]); Serial.print("\t");
                   Serial.print(CoreXY_Pos[2]);     Serial.print("\t");
Serial.print(CoreXY_Pos[3]);

    // Serial.print("V10,V20,V13,V23\t");
          Serial.print("\t");   Serial.print(V10);   Serial.print("\t");
Serial.print(V20);
          Serial.print("\t");   Serial.print(V13);   Serial.print("\t");
Serial.println(V23);

          //           Serial.print("Vdip:\t");   Serial.print(Vdip[0]);
Serial.print("\t"); Serial.print(Vdip[1]);
            //             Serial.print("\t");   Serial.print(Vdip[2]);
Serial.print("\t");           Serial.print(Vdip[3]);Serial.print("\t");
Serial.println(Vdip[4]);
  //  delay(100);
  //}
}



String ReadSerialCommand() {
     Serial.println("Please,  enter  any  command.  Check  -help  for
instructions");
  Serial.println("IDLE");
  String cmd = "";
  while (cmd == "") {
    if (Serial.available()) {
      cmd = Serial.readStringUntil('\n');
    }
  }
  return cmd;
}

void ParseParameters(String* parameters, String command) {
  int index = 0; int lastIndex = 0;
```

```
  int paramNum = 0;
  while ((index != -1) and (paramNum < (int)command.length())) {
    lastIndex = index;
    index =  command.indexOf(' ', index + 1);

    if (paramNum != 0) lastIndex++;
    parameters[paramNum] = command.substring(lastIndex, index);
    // Last parameter
    if (index == -1) {
      parameters[paramNum] = command.substring(lastIndex);
    }
    //    Serial.print("li:");
    //    Serial.println(lastIndex);
    //    Serial.print("i:");
    //    Serial.println(index);
    //    Serial.print("pN:");
    //    Serial.println(paramNum);
    //    Serial.print("param(paramNum):");
    //    Serial.println(parameters[paramNum]);
    paramNum++;
  }
  return;
}

bool checkEndStop(int endStopSIG)
{
  bool sensorValue;
  sensorValue = digitalRead(endStopSIG);

  return sensorValue;
}


void moveMotor(String* parameters) {
     if   ((parameters[1]   ==   "X")   or   (parameters[1]   ==   "Y")   or
(parameters[1] == "Z")) {
         moveStepper(parameters[1].charAt(0),  parameters[2].charAt(0),
parameters[3].toFloat(), false);
  }
  else if (parameters[1] == "A") {
    moveServo(parameters[1].charAt(0), parameters[2].toInt());
  }
  return;
}

void moveServo(char Axis, int Position) {

  //      moveServo('Z', 1);
  //      moveServo('Z', 45);
  //      moveServo('Z', 120);

  Position = max(0, Position);
  Position = min(179, Position);
  int pos = 0;
```

```
  int sign = 1;

  switch (Axis) {
    //    case 'Z':
        //            sign  =  ((Position-CoreXY_Pos[2])/abs((Position-
CoreXY_Pos[2])));
     //       for (pos = CoreXY_Pos[2]; pos != Position; pos += sign)
{ // goes from Actual position to new one
    //        myservoZ.write(pos);
    //        delay(ServoSpeedZ);
    //      }
     //       CoreXY_Pos[2] = Position;           // X(steps), Y(steps,
Z(steps), A(deg, 1-180)
    //
    //      break;
    case 'A':
            sign  =  ((Position  -  CoreXY_Pos[3])  /  abs((Position  -
CoreXY_Pos[3])));
       for (pos = CoreXY_Pos[3]; pos != Position; pos += sign ) { //
goes from Actual position to new one
        myservoA.write(pos);
        delay(ServoSpeedA);
      }
      CoreXY_Pos[3] = Position;          // X(steps), Y(steps, Z(steps),
A(deg, 1-180)
      break;
    default:
      Serial.println("Error: Axis not valid");
      break;
  }

  return;
}


void moveStepper(char Axis, char Direction, long Steps, bool homing) {

  bool DirL = HIGH; // Left Stepper Direction
  bool DirR = HIGH; // Right Stepper Direction
  bool DirZ = HIGH; // Upper Stepper Direction

  long MaxSteps = Steps; // Standard no dimension-limited case


  switch (Axis) {
    case 'X':
      if (Direction == '+') {
        DirL = LOW;
        DirR = LOW;

         if ((CoreXY_Pos[0] + Steps >= Aquarium_stepsX) and (homing ==
false)) { // If we try to exceed the Aquarium dimensions
           MaxSteps = Aquarium_stepsX - CoreXY_Pos[0]; // we just take
the remaining steps
```

```
          Serial.println("End of X reached (max)");
        }
        CoreXY_Pos[0] = CoreXY_Pos[0] + MaxSteps;

      } else if (Direction == '-') {

        DirL = HIGH;
        DirR = HIGH;

        if ((CoreXY_Pos[0] - Steps <= 0) and (homing == false)) { // If
we try to exceed the Aquarium dimensions
           MaxSteps = CoreXY_Pos[0];      // we just take the remaining
steps
          Serial.println("End of X reached (0)");
        }
         CoreXY_Pos[0] = CoreXY_Pos[0] - MaxSteps;         // X(steps),
Y(steps, Z(steps), A(deg, 1-180)
      } else {
        Serial.println("Error: Direction invalid, please use + or -");
        return;
      }
      break;
    case 'Y':
      if (Direction == '+') {
        DirL = LOW;
        DirR = HIGH;

         if ((CoreXY_Pos[1] + Steps >= Aquarium_stepsY) and (homing ==
false)) { // If we try to exceed the Aquarium dimensions
           MaxSteps = Aquarium_stepsY - CoreXY_Pos[1]; // we just take
the remaining steps
          Serial.println("End of Y reached (max)");
        }
         CoreXY_Pos[1] = CoreXY_Pos[1] + MaxSteps;         // X(steps),
Y(steps, Z(steps), A(deg, 1-180)
      } else if (Direction == '-') {

        DirL = HIGH;
        DirR = LOW;

        if ((CoreXY_Pos[1] - Steps <= 0) and (homing == false)) { // If
we try to exceed the Aquarium dimensions
           MaxSteps = CoreXY_Pos[1];      // we just take the remaining
steps
          Serial.println("End of Y reached (0)");
        }
         CoreXY_Pos[1] = CoreXY_Pos[1] - MaxSteps;         // X(steps),
Y(steps, Z(steps), A(deg, 1-180)
      } else {
        Serial.println("Error: Direction invalid, please use + or -");
        return;
      }
      break;
```

```
    case 'Z':
      if (Direction == '+') {
        DirZ = HIGH;

         if ((CoreXY_Pos[2] + Steps >= Aquarium_stepsZ) and (homing ==
false)) { // If we try to exceed the Aquarium dimensions
           MaxSteps = Aquarium_stepsZ - CoreXY_Pos[2]; // we just take
the remaining steps
          Serial.println("Top of Z reached");
        }
        CoreXY_Pos[2] = CoreXY_Pos[2] + MaxSteps;

      } else if (Direction == '-') {

        DirZ = LOW;

        if ((CoreXY_Pos[2] - Steps <= 0) and (homing == false)) { // If
we try to exceed the Aquarium dimensions
           MaxSteps = CoreXY_Pos[2];      // we just take the remaining
steps
          Serial.println("Bottom of Z reached");
        }
         CoreXY_Pos[2] = CoreXY_Pos[2] - MaxSteps;         // X(steps),
Y(steps, Z(steps), A(deg, 1-180)
      } else {
        Serial.println("Error: Direction invalid, please use + or -");
        return;
      }
      break;

    default:
      Serial.println("Error: Axis not valid");
      return;
  }

   digitalWrite(dirPinL, DirL); // Enables the motor to move in a
particular direction
   digitalWrite(dirPinR, DirR); // Enables the motor to move in a
particular direction
   digitalWrite(dirPinZ, DirZ); // Enables the motor to move in a
particular direction

  if ((Axis == 'X') or (Axis == 'Y')) {
    for (int x = 0; x < MaxSteps; x++) {
      digitalWrite(stepPinL, HIGH);
      digitalWrite(stepPinR, HIGH);
      delayMicroseconds(stepSpeed);

      digitalWrite(stepPinL, LOW);
      digitalWrite(stepPinR, LOW);
      delayMicroseconds(stepSpeed);
    }
  } else if (Axis == 'Z')
  {
```

```
    for (int x = 0; x < MaxSteps; x++) {
      digitalWrite(stepPinZ, HIGH);
      delayMicroseconds(stepSpeedZ);

      digitalWrite(stepPinZ, LOW);
      delayMicroseconds(stepSpeedZ);
    }
  }

  return;
}

void move_home(bool deviation = false) {
  // Steppers
  bool HomeX = false;
  bool HomeY = false;
  bool HomeZ = false;

  long Xdif = CoreXY_Pos[0];
  long Ydif = CoreXY_Pos[1];
  long Zdif = CoreXY_Pos[2];

  while (HomeY == false) {
    moveStepper('Y', '-', 1, true);
    if (checkEndStop(endStopYSIG)) {
      HomeY = true;
      CoreXY_Pos[1] = 0;
    }
    Ydif = Ydif - 1;
  }

  while (HomeX == false) {
    moveStepper('X', '-', 1, true);
    if (checkEndStop(endStopXSIG)) {
      HomeX = true;
       CoreXY_Pos[0] = 0;        // X(steps), Y(steps), Z(steps), A(deg,
1-180)
    }
    Xdif = Xdif - 1;
  }

  while (HomeZ == false) {
    moveStepper('Z', '-', 1, true);
    if (checkEndStop(endStopZSIG)) {
      HomeZ = true;
       CoreXY_Pos[2] = 0;        // X(steps), Y(steps), Z(steps), A(deg,
1-180)
    }
    Zdif = Zdif - 1;
  }

  if (deviation == true) {
    Serial.print("ΔX steps :\t"); Serial.println(Xdif);
    Serial.print("ΔY steps :\t"); Serial.println(Ydif);
```

```
    Serial.print("ΔZ steps :\t"); Serial.println(Zdif);
  } else {
    Serial.println("Home position located");
  }
  return;
}

void run_test(int cycles = 0) {
  int i = 0;
  long XtotalSteps = 0;
  long YtotalSteps = 0;
  long ZtotalSteps = 0;

  moveStepper('Z', '-', (Aquarium_stepsZ), false);
  moveStepper('Z', '+', (Aquarium_stepsZ * 0.2), false);

  moveStepper('Y', '-', (Aquarium_stepsY), false);
  moveStepper('X', '-', (Aquarium_stepsX), false);
  moveStepper('Y', '+', (Aquarium_stepsY / 8), false);
  moveStepper('X', '+', (Aquarium_stepsX / 8), false);


       Serial.print("Number    of    cycles    for    the    test:    ");
Serial.println(cycles);
  for (i = 0; i < cycles; i++) {

    moveStepper('X', '+', ((Aquarium_stepsX * 3) / 4), false);
    moveStepper('Y', '+', ((Aquarium_stepsY * 3) / 4), false);
    moveStepper('Z', '+', (Aquarium_stepsZ * 0.6), false);

    moveStepper('X', '-', ((Aquarium_stepsX * 3) / 4), false);
    moveStepper('Y', '-', ((Aquarium_stepsY * 3) / 4), false);
    moveStepper('Z', '-', (Aquarium_stepsZ * 0.6), false);
    Serial.print("Cycle: "); Serial.println(i);
  }

   XtotalSteps = (Aquarium_stepsZ * 0.2) + (Aquarium_stepsX * 1.2) *
cycles;
   YtotalSteps = (Aquarium_stepsY / 8) + (Aquarium_stepsY * 3 / 2) *
cycles;
   ZtotalSteps = (Aquarium_stepsZ / 8) + (Aquarium_stepsZ * 3 / 2) *
cycles;

  move_home(true);
  Serial.print("Total X steps:"); Serial.println(XtotalSteps);
  Serial.print("Total Y steps:"); Serial.println(YtotalSteps);
  Serial.print("Total Z steps:"); Serial.println(ZtotalSteps);

}



void setup() {
```

```
  Serial.begin(115200);
  Serial.println("AQUARIDE V1.0");

  // Steppers
  // Sets the stepper pins as Outputs
  pinMode(stepPinL, OUTPUT);
  pinMode(dirPinL, OUTPUT);

  pinMode(stepPinR, OUTPUT);
  pinMode(dirPinR, OUTPUT);

  pinMode(stepPinZ, OUTPUT);
  pinMode(dirPinZ, OUTPUT);

  pinMode(enablePin, OUTPUT);
  digitalWrite(enablePin, LOW);

  // Servo
   myservoA.attach(servoPinA);   // attaches the servo on pin 19 to the
servo object


  // Endstops
  pinMode(endStopXSIG, INPUT_PULLUP);


  pinMode(endStopYSIG, INPUT_PULLUP);

  pinMode(endStopYGND, OUTPUT);
  digitalWrite(endStopYGND, LOW);

  pinMode(endStopZSIG, INPUT_PULLUP);

  pinMode(endStopZGND, OUTPUT);
  digitalWrite(endStopZGND, LOW);

  // pinMode(endStopXGND, OUTPUT);
  // digitalWrite(endStopXGND, LOW); // Connected to a GND pin


  // Signal receivers

  pinMode(Probe0, OUTPUT);
  digitalWrite(Probe0, LOW);

  pinMode(Probe1, INPUT);
  pinMode(Probe2, INPUT);

  pinMode(Probe3, OUTPUT);
  digitalWrite(Probe3, LOW);

}

void loop() {
```

```
  String command = "";
  command = ReadSerialCommand();

  Serial.print("\n");
  Serial.print("Command:");
  Serial.println(command);

  // String array to store parameters
  String parameters[command.length()];
  for (int i = 0; i < (int)command.length(); i++) {
    parameters[i] = "";
  }

  ParseParameters(parameters, command);
  //Serial.println(parameters[i]);
  if (parameters[0] == "move") {
    Serial.println("Moving");
    moveMotor(parameters);
  }
  else if (parameters[0] == "home") {
    Serial.println("Homing");
    move_home(false);
  }
  else if (parameters[0] == "position") {
    Serial.print("X(steps): \t"); Serial.println(CoreXY_Pos[0]);
    Serial.print("Y(steps):\t"); Serial.println(CoreXY_Pos[1]);
    Serial.print("Z(steps):\t"); Serial.println(CoreXY_Pos[2]);
    Serial.print("A(deg):\t"); Serial.println(CoreXY_Pos[3]);
  }
  else if (parameters[0] == "test") {
    run_test(parameters[1].toInt());
  }
  else if (command == "-help") {
    Serial.println("Operations Implemented:");
    Serial.println("-> home");
    Serial.println("-> sample [Number of samples]");
    Serial.println("-> position");
    Serial.println("-> test [cycle number]");
    Serial.println("-> move [AXIS (X,Y,Z)] [DIRECTION(+,-)] [STEPS]");
    Serial.println("-> move [AXIS (A)]  [POSITION(0,179)]");
  }
  else if (parameters[0] == "sample") {
    GetVoltages(parameters[1].toInt());
  }
  else {
     Serial.println("Error: Unsupported operation, for instructions use
-help");
  }
}
```

F. Código utilizado: OpenSCAD

Cabo_Hilos.scad

```
module Cabo_Hilos(){
    difference(){
        cylinder(h=15,r=8/2,$fn=30);
        translate([0,0,8])
            difference(){
                cylinder(h=4,r=5,$fn=15,center=true);
                cylinder(h=4,r=2.5,$fn=15,center=true);
        }

    }
}

Cabo_Hilos();
```

Carrier.scad

```
use <Eje_Transmision.scad>;
use <Cabo_Hilos.scad>;
use <Carrier_Wheel.scad>;
use <Mirror.scad>;



$fn=30;
radius_bar=5.6;
margen_bar=1;
Radio_eje_rueda=2;
Dims_rueda=[12, 22 ,12] ;
Dist_bar=40; //distancia de una varilla al centro
Posicion_Poleas=[20,45,-20];
Diam_eje_rotatorio=23;
Posicion_ruedas=[Dist_bar,40,-20];
Separacion_receptor_cables=28;

Imprimir = false;

module bar_holder(){
rotate([90,0,0])
    difference(){
        union(){
                cylinder(r=radius_bar+5,h=10,center=true);
            translate([0,10,0])
                cube([(radius_bar+5)*2,20,10],center=true);
        }
        cylinder(r=radius_bar+margen_bar,h=11,center=true);

    }
}

module servo(){
    union(){
        difference(){
            color("white")
                rotate([0,0,90])
                    translate([65,-65,0])
                        import("./Stl/Servo_stand.stl",center = true);
           translate([-40,-25,0])
            cube(30,20,20,center=true);

            mirrorX(Pos=[-27.5,20,0])
            cube(30,20,20,center=true);



        }
   *color("gray")
     rotate([0,0,90])
```

```
        translate([0,-0,2])
            import("./Stl/Futaba3003.stl",center = true);
    }
}

module Carrier(){
    union(){
        difference(){
            cube([102,110,10],center=true);

            // Hueco central
            cylinder(h=30,r=Diam_eje_rotatorio/2,$fn=50,center=true);
        }
        // Añadir poleas
        color("orange")
        mirrorXY(Pos=Posicion_Poleas)
            Cabo_Hilos();

        // Añadimos los amarres
        mirrorXY(Pos=Posicion_ruedas)
            bar_holder();


        // Añadir Eje_transmision
        *translate([0,0,25.6])
            rotate([0,0,-20])
            Eje_Transmision();

        // Añadir servo Alpha
        translate([0,45,5])
            rotate([0,0,0])
            servo();

        // Añadir Rueda Servo Alpha
        *color("teal")
        translate([-0,55,51])
            //Módulo declarado en Eje_transmision.scad
            Rueda_Servo_Alpha();

        // Añadir Receptor de cables
        translate([0,38,0])
        mirrorY(Pos=[-40,Separacion_receptor_cables/2,10])
            cube([15,5,20],center=true);

        translate([-40,38,20])
        rotate([0,-60,0])
            difference(){
                cylinder(h=10,r=28/2,center=true);
                cylinder(h=11,r=22/2,center=true);
            }

    }
}
```

```
Carrier();
*bar_holder();
```

Eje_Transmision_v2.scad

```
use<gears.scad>;
use<Mirror.scad>;
use<publicDomainGear.scad>;
use<Probe.scad>;
use <motormountshelfclip.scad>;


Imprimir=false;

EngranajeZ_Num_Dientes=9;
EngranajeZ_Grosor=8;

Modulo_transmision_EngranajeZ=4;
Modulo_transmision_Servo_Z=10;
Modulo_transmision_Eje_Z_Dentado=8;


Diam_eje_rotatorio=23;
Diam_tope=Diam_eje_rotatorio+5;
Diam_Eje_Amplificadoras=5;
Diam_Plataforma_sup=45;
Diam_Boca_Servo=22;
Diam_eje_motor=6;
Grosor_Plataforma_sup=10;

Altura_Tope_Eje_Z=42; // Establecer en base al tamaño del motor
Altura_Estructura_Central=100;
Altura_Eje_Z_Dentado=Altura_Estructura_Central+420; ////////////////////
250;
Altura_Soporte_Amplificadoras=48;

Altura_Enganche_Eje_Z=20;
Grosor_Enganche_Eje_Z=2;

Ancho_Soporte_Amplificadoras=15;
Ancho_Plataforma_Sup=55;
Ancho_Rueda_Servo_Z=13;
Ancho_Rueda_Amplificadora_Grande=13;
Ancho_Rueda_Amplificadora_Pequena=12;
Ancho_Tubo_Eje_Z=4;
Ancho_Eje_Z=4;
Ancho_pasadizo_alim=4;
Ancho_AlineadorZ=10;

Altura_Eje_Amplificadoras=Ancho_Plataforma_Sup+15;

Posicion_X_Amplificadoras=27.5;
Margen_Holgura_Rotacion=2;
Margen_Encaje_Eje_Z=0.5;
```

```
Diam_eje_Z_dentado=Diam_eje_rotatorio-Margen_Holgura_Rotacion-
Ancho_Tubo_Eje_Z;

module Eje_Z(){
       difference(){
                cylinder(h=Altura_Eje_Z_Dentado,d = Diam_eje_Z_dentado,
$fn=50,center=true);
            translate([Diam_eje_rotatorio-Diam_eje_rotatorio/3.5,0,0])
                cube([Diam_eje_rotatorio,Diam_eje_rotatorio,Altura_Eje_
Z_Dentado],center=true);
        }

}

module Eje_Z_dentado(Print=false){
    if(Print == false){
        difference(){
            union(){
                intersection(){
                    Eje_Z();

                    translate([3,0,Altura_Eje_Z_Dentado])
                    rotate([0,90,-90])
                                                                rack
(mm_per_tooth=(Modulo_transmision_Servo_Z),
                         number_of_teeth=Altura_Eje_Z_Dentado*2/Modulo_t
ransmision_Servo_Z,
                         thickness=Diam_eje_rotatorio, height=20,
                         pressure_angle =0,
                         backlash=0.1 );

                }

                    difference(){
                    translate([5,0,-Altura_Eje_Z_Dentado/2-15/2])
                    cube([25,15,15],center = true);
                    translate([11,0,-Altura_Eje_Z_Dentado/2-29])
                        scale([1.1,1.1,1])
                        probe_connector();

                    scale([1.5,1,1])
                    translate([10,0,-Altura_Eje_Z_Dentado/2-19])
                    probe_connector(tube=true);

                    }
            }

            // Medio cilindro para "ahuecar" el eje
            difference(){
                cylinder(h=Altura_Eje_Z_Dentado+50,d=Diam_eje_Z_dentado
-Ancho_Eje_Z,center=true);

                translate([Diam_eje_Z_dentado/2,0,0])
```

```
                cube([Diam_eje_Z_dentado,Diam_eje_Z_dentado,Altura_Eje_
Z_Dentado+50],center=true);
            }
        }


    }
    else{
         // Partir la pieza por un tercio y ponerla una al lado de la
otra para facilitar impresión
        difference(){
            translate([0,-Altura_Eje_Z_Dentado/6,0])
            difference(){
                translate([0,-Altura_Enganche_Eje_Z,0])
                rotate([90,0,0])
                    Eje_Z_dentado(Print=false);
               translate([0,-Altura_Eje_Z_Dentado/3,0])
                    cube([Altura_Eje_Z_Dentado,Altura_Eje_Z_Dentado,Alt
ura_Eje_Z_Dentado],center=true);
            }
            rotate([90,0,0])
            difference(){
                cylinder(d=Diam_eje_Z_dentado+1,h=Altura_Enganche_Eje_Z
*2,center=true);
                    cylinder(d=Diam_eje_Z_dentado-Grosor_Enganche_Eje_Z-
Margen_Encaje_Eje_Z/2-0.5,h=Altura_Enganche_Eje_Z*2+1,center=true);
            }
        }

        // Pieza central de las 3
        mirror([0,1,0])
        translate([Diam_eje_Z_dentado*3,-Altura_Eje_Z_Dentado/6,0])
        difference(){
            rotate([90,0,0])
                Eje_Z_dentado(Print=false);
            translate([0,Altura_Eje_Z_Dentado/3+Altura_Eje_Z_Dentado/8,
0])
                cube([Altura_Eje_Z_Dentado,Altura_Eje_Z_Dentado/3+Altur
a_Eje_Z_Dentado/4,Altura_Eje_Z_Dentado/3],center=true);
                                    translate([0,-Altura_Eje_Z_Dentado/3-
Altura_Eje_Z_Dentado/8,0])
                cube([Altura_Eje_Z_Dentado,Altura_Eje_Z_Dentado/3+Altur
a_Eje_Z_Dentado/4,Altura_Eje_Z_Dentado/3],center=true);

                                                        translate([0,-
Altura_Eje_Z_Dentado/6+Altura_Enganche_Eje_Z/2,0])
            rotate([90,0,0])
                                            cylinder(d=Diam_eje_Z_dentado-
Grosor_Enganche_Eje_Z+Margen_Encaje_Eje_Z/2,h=Altura_Enganche_Eje_Z,cen
ter=true);
       translate([0,+Altura_Eje_Z_Dentado/6-Altura_Enganche_Eje_Z/2,0])
            rotate([90,0,0])
```

```
                                            cylinder(d=Diam_eje_Z_dentado-
Grosor_Enganche_Eje_Z+Margen_Encaje_Eje_Z/2,h=Altura_Enganche_Eje_Z,cen
ter=true);

        }


        // Tercera pieza
        mirror([0,1,0])
        translate([Diam_eje_Z_dentado*6,0,0])
        difference(){
            translate([0,+Altura_Eje_Z_Dentado/6,0])
            difference(){
                translate([0,+Altura_Enganche_Eje_Z,0])
                rotate([90,0,0])
                    Eje_Z_dentado(Print=false);
               translate([0,+Altura_Eje_Z_Dentado/3,0])
                    cube([Altura_Eje_Z_Dentado,Altura_Eje_Z_Dentado,Alt
ura_Eje_Z_Dentado],center=true);
            }
            rotate([90,0,0])
            difference(){
                cylinder(d=Diam_eje_Z_dentado+1,h=Altura_Enganche_Eje_Z
*2,center=true);
                    cylinder(d=Diam_eje_Z_dentado-Grosor_Enganche_Eje_Z-
Margen_Encaje_Eje_Z/2-0.5,h=Altura_Enganche_Eje_Z*2+1,center=true);
            }
        }
    }
}

module Servo_Futaba_s3003(){
    union(){
        difference(){
            color("white")
                rotate([0,0,90])
                    translate([65,-65,0])
                        import("./Stl/Servo_stand.stl",center = true);
           translate([-40,-25,0])
            cube(30,20,20,center=true);
        }
        if(Imprimir==false){
        color("gray")
            rotate([0,0,90])
            translate([0,-0,2])
                import("./Stl/Futaba3003.stl",center = true);
        }
    }
}

module EjeAmplificadoraGrande(){
                cylinder(h=Altura_Eje_Amplificadoras,d=Diam_Eje_Amplifi
cadoras, $fn=40,center=true);
}
```

xxx

```
module Soporte_Eje_Amplificadora(){
    difference(){
                    union(){
                        cube([Ancho_Soporte_Amplificadoras,Ancho_Soport
e_Amplificadoras,Altura_Soporte_Amplificadoras],center=true);
                                                translate([0,
(Ancho_Soporte_Amplificadoras+Ancho_Rueda_Amplificadora_Grande)/2,3*Alt
ura_Soporte_Amplificadoras/4])
                         cube([Ancho_Soporte_Amplificadoras,2*Ancho_Sop
orte_Amplificadoras+Ancho_Rueda_Amplificadora_Grande,Altura_Soporte_Amp
lificadoras/2],center=true);
                    }
        // Hueco para el eje
                                translate([0,Altura_Eje_Amplificadoras/2-
Ancho_Soporte_Amplificadoras,32-Altura_Soporte_Amplificadoras/2])
        rotate([90,0,0])
        EjeAmplificadoraGrande();

         // Hueco para la rueda

        translate([0,
                    Ancho_Soporte_Amplificadoras/2+Ancho_Rueda_Amplific
adora_Grande/2,
                    32-Altura_Soporte_Amplificadoras/2])
        rotate([90,-5,0])
        cylinder(h=Ancho_Rueda_Amplificadora_Grande+2,d=63,center=true)
;

    }

  }

module Amplificadora_servo_z(){
    union(){
    //PequeñaAmplificadora
        translate([0,12,0])
        rotate([90,70,0])
            gear (
                    mm_per_tooth     = Modulo_transmision_Eje_Z_Dentado,
number_of_teeth    =6,            thickness                          =
Ancho_Rueda_Amplificadora_Pequena,           hole_diameter            =
Diam_Eje_Amplificadoras,    twist           = 0,    teeth_to_hide    = 0,
pressure_angle   = 28,   clearance        = 0.1,   backlash         = 0.0
);

    //GrandeAmplificadora
        rotate([90,32,0])
            gear (
                        mm_per_tooth      = Modulo_transmision_Servo_Z,
number_of_teeth          =15,                                thickness
=Ancho_Rueda_Amplificadora_Grande,           hole_diameter            =
Diam_Eje_Amplificadoras,    twist           = 0,    teeth_to_hide    = 0,
pressure_angle   = 28,   clearance        = 0.0,   backlash         = 0.0
);
```

```
        if(Imprimir==false){
                                translate([0,Altura_Eje_Amplificadoras/2-
2*Ancho_Soporte_Amplificadoras,0])
            rotate([90,0,0])
                EjeAmplificadoraGrande();

        }
    }
}
module Rueda_Stepper_Z(){
difference(){
         gear (
                        mm_per_tooth        =  Modulo_transmision_Servo_Z,
number_of_teeth  =10,      thickness             =  Ancho_Rueda_Servo_Z,
hole_diameter      =  Diam_eje_motor,       twist                    =  0,
teeth_to_hide    = 0,   pressure_angle   = 28,   clearance         = 0.0,
backlash         = 0.0  );

    }
}
module Rueda_Servo_Alpha(){
    difference(){
                            gear_bevel(m  =  Modulo_transmision_EngranajeZ,
//dientes/mm
                              z =EngranajeZ_Num_Dientes*2,     // Numero de
dientes
                           x =0,      //Profile shift
                           h = EngranajeZ_Grosor,    // Altura de la rueda
                           w = 25,    // angulo de los dientes
                           w_bevel = 0, // Ángulo de la rueda
                           w_helix = 0,    //angulo de los dientes
                           D = 0,          //??
                           clearance = -0.1,  // Hueco entre ruedas
                           center = true
                           );
                // Hueco para servo
             translate([0,0,-4])
                cylinder(h=8,d=Diam_Boca_Servo,center=true);
        }

}

module Eje_Transmision_Core(){

    difference(){

        union(){
            // Rueda dentada para permitir giro en alfa
            rotate([0,0,180])
            translate([0,0,Altura_Tope_Eje_Z/2+EngranajeZ_Grosor/2])
                            gear_bevel(m  =  Modulo_transmision_EngranajeZ,
//dientes/mm
```

```
                                                z =EngranajeZ_Num_Dientes,
// Numero de dientes
                                               x =0,     //Profile shift
                                               h = EngranajeZ_Grosor,   //
Altura de la rueda
                                                w = 25,   // angulo de los
dientes
                                                 w_bevel = 0, // Ángulo de
la rueda
                                                w_helix = 0,   //angulo de
los dientes
                                               D = 50,          //??
                                               clearance = -0.1,  // Hueco
entre ruedas
                                               center = true );
            // Tope del eje con <carrier>
                              cylinder(h=Altura_Tope_Eje_Z,d  =  Diam_tope,
$fn=50,center=true);

            translate([0,0,-13.5])
            cylinder(h=15,d2=Diam_tope,d1=Diam_tope+15,center=true);

            // Eje para permitir movimiento en A
                               cylinder(h=Altura_Estructura_Central,d  =
Diam_eje_rotatorio-Margen_Holgura_Rotacion,$fn=50,
    center=true);

            // Plataforma superior para sujección del servo Z

            translate([0,0,(Altura_Tope_Eje_Z/2)+EngranajeZ_Grosor])
                              cylinder(h  =  (Altura_Estructura_Central-
Altura_Tope_Eje_Z)/2-EngranajeZ_Grosor,   d1   =   Diam_tope,   d2   =
Diam_Plataforma_sup, center = true/false);


            // Plataforma para servo superior
            difference(){
                              translate([21,19,Altura_Estructura_Central/2-
Grosor_Plataforma_sup/2])
                     cube([38,Ancho_Plataforma_Sup,Grosor_Plataforma_sup
],center=true);

            }



            translate([20,15,-1+(Altura_Estructura_Central/2)+23])
            rotate([90,0,0])
            nema17shaftplate(40, 45, 8);

            // Alineador EjeZ
           color("teal")
            translate([-Diam_eje_Z_dentado/2-Ancho_AlineadorZ/2+4,0,-1+
(Altura_Estructura_Central/2)+(Altura_Soporte_Amplificadoras/2)])
```

```
            cube([Ancho_AlineadorZ,Ancho_AlineadorZ,Altura_Soporte_Ampl
ificadoras],center=true);
        }


        scale([1.04,1.04,1])
            Eje_Z();

    }

 }



module Pasadizo_Alimentacion(pieza="tapa"){

    if (pieza=="hueco"){
        // Pasadizo para alimentacion del servo Z
                translate([0,Diam_eje_rotatorio/2+Diam_Plataforma_sup/4-
1.5,(Altura_Estructura_Central/4+0.5)])
                 cube([Ancho_pasadizo_alim,Diam_Plataforma_sup/2,Altura_
Estructura_Central/2+1],center=true);
    }
    if  (pieza=="tapa"){
        intersection(){
            Eje_Transmision_Core();
            translate([0,2,3])
            Pasadizo_Alimentacion(pieza="hueco");
        }
    }

}

module Cilindro_Sujeccion_Vertical(){
    difference(){
            cylinder(d=Diam_eje_rotatorio+50,h=8,center=true);

                                            cylinder(d=Diam_eje_rotatorio-
Margen_Holgura_Rotacion+1,h=20,center=true);
    }
}

module Endstop_Z(){
    difference(){
    union(){
        translate([20,0,117.5])
            cube([40,40,15],center=true);
        translate([0,0,120])
            cylinder(r=20,h=10,center=true);
    }
    scale([1.02,1.02,1])
    Eje_Z();

    translate([-15,0,117.5])
```

```
        cube([15,4,15],center=true);
    }
}
module Eje_Transmision(){
    if(Imprimir==false){

        union(){
            difference(){
                Eje_Transmision_Core();
                Pasadizo_Alimentacion(pieza="hueco");
            }
            Pasadizo_Alimentacion(pieza="tapa");

        }

        //Eje dentado para transmisión en Z
        color("yellow")
        translate([0,0,-150])
        render()
            Eje_Z_dentado();

        //Rueda stepper en Z
        *color("green")
            translate([20,0,22+Altura_Estructura_Central/2])
            rotate([90,19,0])
            render()
                Rueda_Stepper_Z();


        // Nema17
        %color("teal")
        translate([20,20.5,22+Altura_Estructura_Central/2])
        rotate([90,0,0])
            import("./Stl/NEMA17.stl");



        // Sistema de engranajes
          *color("orange")
            translate([Posicion_X_Amplificadoras,0,32+Altura_Estructura
_Central/2])
             render()
                    Amplificadora_servo_z();


        //Rueda acoplada a servo en alpha
        *translate([60,0,25])
        rotate([0,180,0])
                Rueda_Servo_Alpha();

        //Rueda para estabilizar ejeZ
        *translate([0,0,-40])
        Cilindro_Sujeccion_Vertical();
```

```
        *Endstop_Z();
    }
    else {

        *difference(){
                Eje_Transmision_Core();
                Pasadizo_Alimentacion(pieza="hueco");
            }

        *scale([0.9,1,0.9])
         Eje_Z_dentado(Print=true);


         *Rueda_Stepper_Z();

         rotate([90,0,0])
         *Amplificadora_servo_z();

         rotate([0,180,0])
         *Rueda_Servo_Alpha();

         *scale([0.7,0.7,1])
         EjeAmplificadoraGrande();

        *Pasadizo_Alimentacion(pieza="tapa");

        *Cilindro_Sujeccion_Vertical();

        *Endstop_Z();
    }
}

Eje_Transmision();
*Endstop_Z();
```

Mirror.scad

```
module mirrorXY(Pos=[10,10,0])
 {
    translate(Pos)
        child();

    mirror([1,0,0])
        translate(Pos)
            child();
    mirror([0,1,0])
        translate(Pos)
           child();

    mirror([1,0,0])
        mirror([0,1,0])
            translate(Pos)
                child();
}

// Módulos
module mirrorX(Pos=[10,10,0])
 {
    translate(Pos)
        child();
    mirror([1,0,0])
        translate(Pos)
            child();
}

module mirrorY(Pos=[10,10,0])
 {
    translate(Pos)
        child();
    mirror([0,1,0])
        translate(Pos)
           child();
}

module mirrorZ(Pos=[10,10,0])
 {
    translate(Pos)
        child();
    mirror([0,1,0])
        translate(Pos)
           child();
}
```

Motor_Wheel.scad

```
$fn=30;
radius_bar =7; //mm
bend_radius =7 ; //long wheel


module Carrier_Wheel_axle(){
    cylinder(r=1.4,h=21,center=true);

}

 module Carrier_Wheel(){

    difference() {
        cylinder(r=12,h=15,center=true);

        rotate_extrude()
        translate([bend_radius + radius_bar, 0, 0])
            circle(r=radius_bar,$fn=7);

        cylinder(d=6,h=20,center=true);
     }
}

Carrier_Wheel();

//Carrier_Wheel_axle();
```

Polea_Alta.scad

```
$fn=20;

radius_bar =3; //mm

bend_radius =4; //long wheel

difference() {
cylinder(r1=15, r2=5,h=25,,center=true);
translate([0,0,4])
    rotate_extrude()
        translate([bend_radius + radius_bar, 0, 0])
            circle(r=radius_bar);
}
```

Polea_baja.scad

```
$fn=20;
radius_bar =2; //mm
bend_radius =4; //long wheel

module Polea_baja(){
        difference() {
            cylinder(r1=8, r2=5,h=8 ,center=true);

            rotate_extrude()
            translate([bend_radius + radius_bar, 0, 0])
                circle(r=radius_bar);
        }
}

Polea_baja();
```

Probe_V2.0.scad

```
use<Mirror.scad>;


$fn=50;


diam_screw_head=15; // Diametro de la tuerca
diam_screw=9;
diam_ext_electrodo=23;
radio_probe=diam_ext_electrodo/2;
height_screw=6;

length_section=5+height_screw*2;
separation=5; // Se puede animar!!
distance_sections=separation+length_section;
number_parts=5;
radio_cable=2;

grosor_separacion_vertical=0;

altura_conector=25;
ancho_conector=7;
anchura_conector_pole=4.3;

module probe_connector(tube=false){

        if (tube==true)
    {
            translate([0,0,altura_conector/2])
    cube([anchura_conector_pole,anchura_conector_pole,altura_conector],
center=true);
    } else{
    translate([0,0,altura_conector/2])
    cube([anchura_conector_pole,anchura_conector_pole,altura_conector],
center=true);

    translate([0,0,altura_conector+ancho_conector/2])
    cube([ancho_conector,ancho_conector,ancho_conector],center=true);
    }

}

module probe(){

difference(){
    union(){
    translate([0,-(distance_sections*(number_parts+1))/2,0])
    for (aux=  [1:number_parts])
        {
            translate([0,aux*distance_sections,0])
            rotate([90,0,0])
```

```
                cylinder(r=radio_probe, h=length_section,center=true);
        }
    }


    translate([0,-(distance_sections*(number_parts))/2,0])
    for (aux=  [0:number_parts]){
         translate([0,aux*distance_sections,0])
            rotate([90,0,0])
                difference(){
                                        cylinder(r=(diam_screw_head+1)/2,
h=height_screw*2+separation,center=true,$fn=6);

                }
        }

}
}
```

```
probe();
translate([0,0,radio_probe/2])
    probe_connector();
```

Top_Corner.scad

```
 // Variables
ancho_cristal=5.5;
ancho_piezamain=8;
size_piezamain=60;
size_enganche=30;
ancho_enganche=8;
size_platf_poleaAlta=40;

module pestanita(){
    hueco_silicona_size=40;

    difference(){
        translate([ancho_cristal,ancho_cristal,-size_enganche])
            cube([size_enganche,size_enganche,size_enganche]);

         translate([ancho_cristal+ancho_enganche,ancho_cristal+ancho_eng
anche,-size_enganche])
            cube([size_enganche,size_enganche,size_enganche]);

                        translate([ancho_cristal/1.5,ancho_cristal/1.5,-
size_enganche/2])
            cylinder(h=size_enganche,d=20,center=true,$fn=4);
        }
}

module Top_corner_motor(){
    union(){
        // PiezaEsquina
        difference(){
          translate([size_piezamain/2-ancho_piezamain,size_piezamain/2-
ancho_piezamain,-(size_piezamain/2-ancho_piezamain)])
            cube([size_piezamain,size_piezamain,size_piezamain],center=
true);
        translate([50,50,-50])
            cube([100,100,100],center=true);
        }

        pestanita();


                                    translate([-size_platf_poleaAlta/5,-
size_platf_poleaAlta/5,ancho_piezamain/2])
            cylinder(r=size_platf_poleaAlta/2,h=ancho_piezamain,center=
true);

        difference(){
           translate([-size_platf_poleaAlta/5,-size_platf_poleaAlta/5,-
ancho_piezamain])
            cylinder(r2=size_platf_poleaAlta/2,h=ancho_piezamain*2,cent
er=true);
        translate([50,50,-50])
```

```
            cube([100,100,100],center=true);

        }

        translate([10,5,12])
            import("./Stl/Polea_baja.stl");

        translate([-10,-10,20])
            import("./Stl/Polea_alta.stl");
    }


}
Top_corner_motor();
```

Top_Corner_Motor.scad

```
use <motormountshelfclip.scad>;

// Variables
ancho_cristal=5.5;
size_soporteMotor=50;
ancho_piezamain=7;
size_piezamain=60;
size_enganche=40;
ancho_enganche=6;

module pestanita(){
    hueco_silicona_size=40;

    difference(){
        translate([ancho_cristal,ancho_cristal,-size_enganche])
            cube([size_enganche,size_enganche,size_enganche]);

         translate([ancho_cristal+ancho_enganche,ancho_cristal+ancho_eng
anche,-size_enganche])
            cube([size_enganche,size_enganche,size_enganche]);

                        translate([ancho_cristal/1.4,ancho_cristal/1.4,-
size_enganche/2])
            cylinder(h=size_enganche,d=20,center=true,$fn=4);
        }
}

module cilindro_eje(){

    //Cilindro para la cuerda
    difference(){

        cylinder(d=10,h=20,center=true);
        color("teal")
        translate([0,0,0.01])
        translate([0,0,-20])
            scale([1.02,1.02,1.01])
                import("./Stl/NEMA17.stl");
        difference(){
            translate([0,0,5])
                cylinder(d=12,h=6,center=true);
            translate([0,0,5])
                cylinder(d=7,h=6,center=true);
        }
    }
}
module Top_corner_motor(){

    union(){
        // PiezaEsquina
        difference(){
```

```
          translate([size_piezamain/2-ancho_piezamain,size_piezamain/2-
ancho_piezamain,-(size_piezamain/2-ancho_piezamain)])
             cube([size_piezamain,size_piezamain,size_piezamain],center=
true);
        translate([50,50,-50])
            cube([100,100,100],center=true);
        }

        pestanita();

        translate([0,-29,2])
        nema17shaftplate(43, 45, 10);

        translate([40,-18,0])
        difference(){
            cylinder(h=10,d=28,center=true);
            cylinder(h=11,d=22,center=true);

            }



        // Nema17
        %color("teal")
        translate([0,-29,-1])
        scale([1.00,1.00,1.00])
            import("./Stl/NEMA17.stl");

    }
}

Top_corner_motor();
*cilindro_eje();
```

Wagon.scad

```
Dist_bar=40; //distancia de una varilla al centro
radius_bar =5.3; //mm
largo_bar=100;
pos_bar=[0,Dist_bar,9];
margen_bar=1;
ancho_cristal=5.5+0.4;
ancho_pestanita=6;

use <Mirror.scad>
use <Polea_baja.scad>

Altura_Base=36;
Anchura_Base= (Dist_bar*2)+20;
Largo_Base=45;

module varilla(){
    translate([largo_bar/2,0,0])
        rotate([0,90,0])
            cylinder(r=radius_bar+margen_bar,  h=largo_bar,center=true,
$fn=100);
}


module wagon(){
    difference(){
        translate([5,0,0])
        cube([Largo_Base,Anchura_Base,Altura_Base],center=true);

        translate([0,-Anchura_Base/2-5,-Altura_Base])
            cube([Largo_Base,Anchura_Base+10,Altura_Base]);

        %mirrorY(pos_bar)
            varilla();

        difference(){
            translate([-Largo_Base-8,-(Anchura_Base+5)/2,0+8])
             cube([Largo_Base,Anchura_Base+5,Altura_Base]);

            translate([-8,0,8])
            rotate([90,0,0])
            cylinder(r=10,h=Anchura_Base+6,center=true);
        }

    }

    mirrorY([5,Dist_bar/2,21])
        Polea_baja();


}
```

```
wagon();
```

G. Código utilizado: Python

Sample5K.py

```
import serial
import time
import sys
import math

baudRate = 115200
serialPort = '/dev/ttyUSB0'
arduino = serial.Serial(serialPort, baudrate=baudRate, timeout=0.2)



def cm2stepsX ( centimeters ):
        steps = math.floor(centimeters * 621.1) #ratio steps/cm
        return steps
def cm2stepsY ( centimeters ):
        steps = math.floor(centimeters * 651.46)
        return steps
def cm2stepsZ ( centimeters ):
        steps = math.floor(centimeters * 304.87)
        return steps

def Read_Aquarride( arduino ):
        while True:
                data = arduino.readline()[:-2] # \n
                if data:
                        print data
                if data == 'IDLE':
                        break
        return

def Parse_Aquarride( arduino, fd ):

        while True:
                data = arduino.readline()[:-2] # \n
                if data[:5]=='DATA:':
                        Data_List = data.split("\t")
                        if len(Data_List)==9:
                                fd.write(Data_List[1]+';'+Data_List[2]+
';'+Data_List[3]+';'+Data_List[4]+';'+Data_List[5]+';'+Data_List[6]+';'
+Data_List[7]+';'+Data_List[8]+'\n')
                        else:
                                print('Warning, data error')
                elif data == 'IDLE':
                        break
        return

# Create data file and open it to append new data
timestr = time.strftime("%Y-%m-%d %H:%M:%S")
```

```
print('Creating data file')
name = 'csv/'+sys.argv[0]+'-'+timestr +'.csv'  # Name of csv file


try:
        fd = open(name,'a')    # Trying to create a new file or open one
         fd.write('POSX;POSY;POSZ;POSA;V10;V20;V13;V23\n') #headers for
the data file

except:
        print('Something went wrong! Can\'t tell what?')
        sys.exit(0) # quit Python


# Manual Reset
arduino.setDTR(False)
time.sleep(1)
arduino.flushInput()
arduino.setDTR(True)


Read_Aquarride( arduino )

# In the fixed position, make 5k measures (x4)

arduino.write('sample 5000')
Parse_Aquarride( arduino, fd )


fd.close()
print 'Experimento finalizado'

sys.exit(0) # quit Python
```

Sweep.py

```
import serial
import time
import sys
import math
import numpy as np

baudRate = 115200
serialPort = '/dev/ttyUSB0'
arduino = serial.Serial(serialPort, baudrate=baudRate, timeout=0.2)

# From Calibration Phase
Aquarium_stepsX = 15300;
Aquarium_stepsY = 13300;
Aquarium_stepsZ = 8300;

def cm2stepsX ( centimeters ):
        steps = math.floor(centimeters * 621.1) #ratio steps/cm
        return steps
def cm2stepsY ( centimeters ):
        steps = math.floor(centimeters * 651.46)
        return steps
def cm2stepsZ ( centimeters ):
        steps = math.floor(centimeters * 304.87)
        return steps

def Read_Aquarride( arduino ):
        while True:
                data = arduino.readline()[:-2] # \n
                if data:

                        print data
                if data == 'IDLE':
                        break
        return

def Parse_Aquarride( arduino, fd ):

        while True:
                data = arduino.readline()[:-2] # \n
                if data[:5]=='DATA:':
                        Data_List = data.split("\t")
                        if len(Data_List)==9:
                                fd.write(Data_List[1]+';'+Data_List[2]+
';'+Data_List[3]+';'+Data_List[4]+';'+Data_List[5]+';'+Data_List[6]+';'
+Data_List[7]+';'+Data_List[8]+'\n')
                        else:
                                print('Warning, data error')
                elif data == 'IDLE':
                        break
        return
```

```
def SweepRead (Axis,samples,NumPositions, fd, arduino):
        # Sweeps selected axis and reads <NumPositions> positions with
indicated number of samples

        if Axis == 'X':
                Aquarium_steps= Aquarium_stepsX
        elif Axis == 'Y':
                Aquarium_steps= Aquarium_stepsY
        elif Axis == 'Z':
                Aquarium_steps= Aquarium_stepsZ
        else:
                print ('Warning in Sweep: axis not valid')

             StepJump = np.linspace(0, Aquarium_steps, NumPositions,
endpoint=True)[1]

                #print ('X range'+str(np.linspace(0, Aquarium_steps,
NumPositions, endpoint=True)))
              for i in np.linspace(0, Aquarium_steps, NumPositions,
endpoint=True):
                print('Sampling at X:'+str(i))
                arduino.write('sample '+ str(samples))
                Parse_Aquarride( arduino, fd )
                  if (i<Aquarium_steps): #Stop moving at the end of the
aquarium
                                arduino.write('move ' + Axis + ' + ' +
str(StepJump))
                        Read_Aquarride( arduino)
                else:
#                       arduino.write('home')
#                       Read_Aquarride( arduino)
                                arduino.write('move ' + Axis + ' - ' +
str(Aquarium_steps))
                        Read_Aquarride( arduino)
        return


# Create data file and open it to append new data
timestr = time.strftime("%Y-%m-%d %H:%M:%S")
print('Creating data file')
name = 'csv/'+sys.argv[0]+'-'+ raw_input('Introduzca nombre para el
experimento:')+'-' +timestr +'.csv'  # Name of csv file

try:
        fd = open(name,'w')    # Trying to create a new file or open one
         fd.write('POSX;POSY;POSZ;POSA;V10;V20;V13;V23\n') #headers for
the data file

except:
        print('Something went wrong! Can\'t tell what?')
        sys.exit(0) # quit Python

# Manual Reset
```

```
arduino.setDTR(False)
time.sleep(1)
arduino.flushInput()
arduino.setDTR(True)

Read_Aquarride( arduino )

# Set home position and move to position 0
arduino.write('home')
Read_Aquarride( arduino )

arduino.write('move A 125')
Read_Aquarride( arduino )

#Position->[0,0,0,50]
#Starting position -> lateral sweep
Z=2000; #height
Y=5500;

arduino.write('move Z + ' + str(Z))
Read_Aquarride( arduino )
arduino.write('move Y + ' + str(Y))
Read_Aquarride( arduino )

SweepRead(Axis='X',samples=300,NumPositions=10, fd=fd, arduino=arduino)

fd.close()
print 'Experimento finalizado'

sys.exit(0) # quit Python
```

2D_Mapping.py

```
import serial
import time
import sys
import math
import numpy as np

baudRate = 115200
serialPort = '/dev/ttyUSB0'
arduino = serial.Serial(serialPort, baudrate=baudRate, timeout=0.2)

# From Calibration Phase
Aquarium_stepsX = 15300;
Aquarium_stepsY = 13300;
Aquarium_stepsZ = 8300;

def cm2stepsX ( centimeters ):
        steps = math.floor(centimeters * 621.1) #ratio steps/cm
        return steps
def cm2stepsY ( centimeters ):
        steps = math.floor(centimeters * 651.46)
        return steps
def cm2stepsZ ( centimeters ):
        steps = math.floor(centimeters * 304.87)
        return steps

def Read_Aquarride( arduino ):
        while True:
                data = arduino.readline()[:-2] # \n
                if data:

                        print data
                if data == 'IDLE':
                        break
        return

def Parse_Aquarride( arduino, fd ):

        while True:
                data = arduino.readline()[:-2] # \n
                if data[:5]=='DATA:':
                        Data_List = data.split("\t")
                        if len(Data_List)==9:
                                fd.write(Data_List[1]+';'+Data_List[2]+
';'+Data_List[3]+';'+Data_List[4]+';'+Data_List[5]+';'+Data_List[6]+';'
+Data_List[7]+';'+Data_List[8]+'\n')
                        else:
                                print('Warning, data error')
                elif data == 'IDLE':
                        break
        return
```

```
def SweepRead (Axis,samples,NumPositions, fd, arduino):
        # Sweeps selected axis and reads <NumPositions> positions with
indicated number of samples

        if Axis == 'X':
                Aquarium_steps= Aquarium_stepsX
        elif Axis == 'Y':
                Aquarium_steps= Aquarium_stepsY
        elif Axis == 'Z':
                Aquarium_steps= Aquarium_stepsZ
        else:
                print ('Warning in Sweep: axis not valid')

             StepJump = np.linspace(0, Aquarium_steps, NumPositions,
endpoint=True)[1]

                #print ('X range'+str(np.linspace(0, Aquarium_steps,
NumPositions, endpoint=True)))
               for i in np.linspace(0, Aquarium_steps, NumPositions,
endpoint=True):
                print('Sampling at X:'+str(i))
                arduino.write('sample '+ str(samples))
                Parse_Aquarride( arduino, fd )
                 if (i<Aquarium_steps): #Stop moving at the end of the
aquarium
                                arduino.write('move ' + Axis + ' + ' +
str(StepJump))
                        Read_Aquarride( arduino)
                else:
#                       arduino.write('home')
#                       Read_Aquarride( arduino)
                                arduino.write('move ' + Axis + ' - ' +
str(Aquarium_steps))
                        Read_Aquarride( arduino)
        return


def SweepReadXY (samples,NumPositionsX,NumPositionsY, Z, fd, arduino):
        # Sweeps plane at current Z

        MaxY=Aquarium_stepsY
         StepJump = np.linspace(0, MaxY, NumPositionsY, endpoint=True)
[1]
        k=1;

        for j in np.linspace(0, MaxY, NumPositionsY, endpoint=True):
                print ('Sweeping X at Y:'+str(j))
                                                        SweepRead
(Axis='X',samples=samples,NumPositions=NumPositionsX,           fd=fd,
arduino=arduino)
                if (j<MaxY): #Stop moving at the end of the aquarium
#                       arduino.write('move Z + '+ str(Z))
#                       Read_Aquarride( arduino )
```

```
                                arduino.write('move ' + 'Y' + ' + ' +
str(StepJump)) # str(StepJump*k))
                        Read_Aquarride( arduino)

                k+=1;
        arduino.write('home')
        Read_Aquarride( arduino)
        return


# Create data file and open it to append new data
timestr = time.strftime("%Y-%m-%d %H:%M:%S")
print('Creating data file')
name = 'csv/'+sys.argv[0]+'-'+ raw_input('Introduzca nombre para el
experimento:')+'-' +timestr +'.csv'  # Name of csv file

try:
        fd = open(name,'w')   # Trying to create a new file or open one
         fd.write('POSX;POSY;POSZ;POSA;V10;V20;V13;V23\n') #headers for
the data file

except:
        print('Something went wrong! Can\'t tell what?')
        sys.exit(0) # quit Python

# Manual Reset
arduino.setDTR(False)
time.sleep(1)
arduino.flushInput()
arduino.setDTR(True)

Read_Aquarride( arduino )

# Set home position and move to position 0
arduino.write('home')
Read_Aquarride( arduino )

arduino.write('move A 125')
Read_Aquarride( arduino )

#Position->[0,0,0,50]
#Starting position -> lateral sweep
Z=4000; #height

arduino.write('move Z + ' + str(Z))
Read_Aquarride( arduino )

SweepReadXY(samples=300,NumPositionsX=20,NumPositionsY=20,  Z=Z,  fd=fd,
arduino=arduino)

fd.close()
print 'Experimento finalizado'

sys.exit(0) # quit Python
```

2D_Explore.py

```
import serial
import time
import sys
import math
import numpy as np

baudRate = 115200
serialPort = '/dev/ttyUSB0'
arduino = serial.Serial(serialPort, baudrate=baudRate, timeout=0.2)

# From Calibration Phase
Aquarium_stepsX = 15300;
Aquarium_stepsY = 13300;
Aquarium_stepsZ = 8300;

def cm2stepsX ( centimeters ):
        steps = math.floor(centimeters * 621.1) #ratio steps/cm
        return steps
def cm2stepsY ( centimeters ):
        steps = math.floor(centimeters * 651.46)
        return steps
def cm2stepsZ ( centimeters ):
        steps = math.floor(centimeters * 304.87)
        return steps

def Read_Aquarride( arduino ):
        while True:
                data = arduino.readline()[:-2] # \n
                #if data:

                        #print data
                if data == 'IDLE':
                        break
        return

def Check_Object(lineSamples):
        flag = False
        FrontFlag=False
        BackFlag=False
        objectThreshold = 8
        if lineSamples.shape[0]>1:
                V10dif=lineSamples[-1,4]-lineSamples[-2,4]
                V20dif=lineSamples[-1,5]-lineSamples[-2,5]
                V13dif=lineSamples[-1,6]-lineSamples[-2,6]
                V23dif=lineSamples[-1,7]-lineSamples[-2,7]

                FrontIndicator= V10dif+V20dif-V13dif-V23dif
                BackIndicator= -V10dif-V20dif+V13dif+V23dif

                 FrontFlag= FrontIndicator>objectThreshold and V10dif>0
and V20dif>0 and V13dif<0 and V23dif<0
```

```
                    BackFlag= BackIndicator>objectThreshold and V10dif<0
and V20dif<0 and V13dif>0 and V23dif>0
                state=''
                if (V10dif>0 and V20dif>0 and V13dif<0 and V23dif<0):
                        state='Approaching front'
                elif (V10dif<0 and V20dif<0 and V13dif>0 and V23dif>0):
                        state='Approaching back'

                          print 'FrontIndicator= '+str(FrontIndicator)
+'\tFrontFlag = '+ str(FrontFlag)
                 print 'BackIndicator= '+str(BackIndicator)+'\tBackFlag
= '+ str(BackFlag)
                print state

                flag=FrontFlag
        return flag


def Parse_Aquarride( arduino, fd ):

        while True:
                data = arduino.readline()[:-2] # \n
                if data[:5]=='DATA:':
                        Data_List = data.split("\t")
                        if len(Data_List)==9:
                                fd.write(Data_List[1]+';'+Data_List[2]+
';'+Data_List[3]+';'+Data_List[4]+';'+Data_List[5]+';'+Data_List[6]+';'
+Data_List[7]+';'+Data_List[8]+'\n')
                                        sampleData=Data_List[1:] #without
"DATA:"
                                         sampleData = np.array(map(float,
sampleData)) #conversion to float (read as string)
                        else:
                                print('Warning, data error')
                elif data == 'IDLE':
                        break
        return sampleData


def SweepRead (Axis,samples,NumPositions, fd, arduino):
         # Sweeps selected axis and reads <NumPositions> positions with
indicated number of samples

        if Axis == 'X':
                Aquarium_steps= Aquarium_stepsX
        elif Axis == 'Y':
                Aquarium_steps= Aquarium_stepsY
        elif Axis == 'Z':
                Aquarium_steps= Aquarium_stepsZ
        else:
                print ('Warning in Sweep: axis not valid')

              StepJump  =  np.linspace(0,  Aquarium_steps,  NumPositions,
endpoint=True)[1]
```

```
        lineSamples=np.empty((0,8))


        k=0

                for i in np.linspace(0, Aquarium_steps, NumPositions,
endpoint=True):
                  print('Sampling at X:'+str(i))
                  arduino.write('sample '+ str(samples))
                  Data=Parse_Aquarride( arduino, fd )
                  lineSamples = np.vstack([lineSamples, Data])
                  Flag_NearObject = Check_Object(lineSamples)


                  if Flag_NearObject==True:
                                  arduino.write('move ' + Axis + ' - ' +
str(Aquarium_steps))
                          Read_Aquarride( arduino)
                          break

                   if (i<Aquarium_steps): #Stop moving at the end of the
aquarium
                                  arduino.write('move ' + Axis + ' + ' +
str(StepJump))
                          Read_Aquarride( arduino)
                  else:
#                         arduino.write('home')
#                         Read_Aquarride( arduino)
                                  arduino.write('move ' + Axis + ' - ' +
str(Aquarium_steps))
                          Read_Aquarride( arduino)
                  k+=1
        return lineSamples


def SweepReadXY (samples,NumPositionsX,NumPositionsY, Z, fd, arduino):
        # Sweeps plane at current Z

            StepJump = np.linspace(0, Aquarium_stepsY, NumPositionsY,
endpoint=True)[1]
        k=1;

              for j in np.linspace(0, Aquarium_stepsY, NumPositionsY,
endpoint=True):
                  #print ('Sweeping X at Y:'+str(j))
                                                                  SweepRead
(Axis='X',samples=samples,NumPositions=NumPositionsX,             fd=fd,
arduino=arduino)
                   if (j<Aquarium_stepsY): #Stop moving at the end of the
aquarium
#                         arduino.write('move Z + '+ str(Z))
#                         Read_Aquarride( arduino )
                                  arduino.write('move ' + 'Y' + ' + ' +
str(StepJump)) # str(StepJump*k))
```

```
                        Read_Aquarride( arduino)

                k+=1;
        arduino.write('home')
        Read_Aquarride( arduino)
        return


# Create data file and open it to append new data
timestr = time.strftime("%Y-%m-%d %H:%M:%S")
print('Creating data file')
name  =  'csv/'+sys.argv[0]+'-'+  raw_input('Introduzca  nombre  para  el
experimento:')+'-' +timestr +'.csv'  # Name of csv file

try:
        fd = open(name,'w')    # Trying to create a new file or open one

except:
        print('Something went wrong! Can\'t tell what?')
        sys.exit(0) # quit Python

# Manual Reset
arduino.setDTR(False)
time.sleep(1)
arduino.flushInput()
arduino.setDTR(True)

Read_Aquarride( arduino )

# Set home position and move to position 0
arduino.write('home')
Read_Aquarride( arduino )

arduino.write('move A 125')
Read_Aquarride( arduino )

#Position->[0,0,0,50]
#Starting position -> lateral sweep
Z=1750; #height

arduino.write('move Z + ' + str(Z))
Read_Aquarride( arduino )

NumPositionsX=15
NumPositionsY=15

fd.write('PositionsExploration:;NumX=;'+str(NumPositionsX)
+';NumY=;'+str(NumPositionsY)+';MaxX=;'+str(Aquarium_stepsX)
+';MaxY=;'+str(Aquarium_stepsY)+'\n') #headers for the data file
fd.write('POSX;POSY;POSZ;POSA;V10;V20;V13;V23\n') #headers for the data
file

SweepReadXY(samples=300,NumPositionsX=NumPositionsX,NumPositionsY=NumPo
sitionsY, Z=Z, fd=fd, arduino=arduino)
```

```
fd.close()
print 'Experimento finalizado'

sys.exit(0) # quit Python
```

CsvPlotter.py

```
import matplotlib.pyplot as plt
import numpy as np
import sys

name = sys.argv[1]  # Name of csv file

try:
        fd = open(name,'r')    # Trying to open

except:
        print('No se pudo abrir el archivo indicado')
        sys.exit(0) # quit Python

dataMatrix=np.loadtxt(fd,delimiter=";",skiprows=1)
fd.close()


plt.plot(dataMatrix[:,4:8])
plt.title("Barrido lateral: Pecera Vacia")
plt.xlabel("Posicion")
plt.ylabel("Valor")
plt.legend(["V10,V20,V13,V23"],loc='best')
plt.show()
sys.exit(0)
```

CsvPlotter2D.py

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import matplotlib.pyplot as plt
import numpy as np
import sys
import time
from mpl_toolkits.axes_grid1 import make_axes_locatable


def steps2cmX ( steps ):
        cm = steps / 621.1
        return cm
def steps2cmY ( steps ):
        cm = steps / 651.46
        return cm
def steps2cmZ ( steps ):
        cm = steps / 304.87
        return cm


name = sys.argv[1]  # Name of csv file

try:
        fd = open(name,'r')    # Trying to open

except:
        print('No se pudo abrir el archivo indicado')
        sys.exit(0) # quit Python

meshData = fd.readline()
meshData = meshData[:-1].split(";")


dataMatrix=np.loadtxt(fd,delimiter=";",skiprows=1)
fd.close()

V=np.empty([int(meshData[2]),int(meshData[4]),4])
MaxX=int(meshData[6])
MaxY=int(meshData[8])

X_index = np.linspace(0, MaxX, meshData[2], endpoint=True)
Y_index = np.linspace(0, MaxY, meshData[4], endpoint=True)

for rows in dataMatrix:
        X_aprox = min(X_index, key=lambda x:abs(x-rows[0]))
        X= np.where(X_index==X_aprox)[0]

        Y_aprox = min(Y_index, key=lambda x:abs(x-rows[1]))
        Y= np.where(Y_index==Y_aprox)[0]
```

```
        V[X,Y,:]=rows[4:]


#NumPosX=len(np.unique(dataMatrix[:,0]))
#NumPosY=len(np.unique(dataMatrix[:,1]))
#V=np.empty([NumPosY,NumPosX,4])



#for i in range(NumPosY):
#       for j in range(NumPosX):
##                                          print   'V['+str(i)+','+str(j)
+']=dataMatrix['+str((NumPosX*i)+j)+',4]'
##                                 print  'dataMatrix['+str((NumPosX*i)+j)
+',4]='+str(dataMatrix[(NumPosX*i)+j,4])
#               V[i,j,:]=dataMatrix[(NumPosX*i)+j,4:]
V=np.rot90(V)

V[V == 0.0] = np.nan
V10=V[:,:,0]
V20=V[:,:,1]
V13=V[:,:,2]
V23=V[:,:,3]
names=['V10','V20','V13','V23']

# specifies the number of rows and columns in the figure. this will
create (row x column) number of subplots.
row = 2
column = 2

# fig refers to the figure that will display the subplots
# ax is an array and refers to the axis for each subplot
fig, ax = plt.subplots(row, column, facecolor='w', figsize=(15,10))

# sets the title to be displayed at the top of the figure.
fig.suptitle(name[27:-9], fontsize=24)

# iterates over each axis, ax, and plots random data
for i, ax in enumerate(ax.flat, start=0):

  # sets the title for subplot i
  ax.set_title(names[i])

  # plots random data using the 'jet' colormap
        img     =     ax.imshow(V[:,:,i],     vmin=np.nanmin(V[:,:,i]),
vmax=np.nanmax(V[:,:,i]),interpolation='none')

   # creates a new axis, cax, located 0.05 inches to the right of ax,
whose width is 15% of ax
  # cax is used to plot a colorbar for each subplot
  div = make_axes_locatable(ax)
  cax = div.append_axes("right", size="15%", pad=0.05)
```

```
                cbar        =        plt.colorbar(img,        cax=cax,
ticks=np.arange(np.nanmin(V[:,:,i]),np.nanmax(V[:,:,i]),1),
format="%.2f")
  cbar.set_label('Colorbar {}'.format(i), size=10)

  # removes x and y ticks
  ax.xaxis.set_visible(False)
  ax.yaxis.set_visible(False)

# prevents each subplot's axes from overlapping
plt.tight_layout()

# moves subplots down slightly to make room for the figure title
plt.subplots_adjust(top=0.9)
plt.show()
```

```
#fig, axes = plt.subplots(nrows=2, ncols=2)
#i=0
#for ax in axes.flat:
#                    im   =   ax.imshow(V[:,:,i],   vmin=V[:,:,:].min(),
vmax=V[:,:,:].max())
#       ax.set_title('V[:,:,'+str(i)+']')
#       i+=1
#       plt.colorbar()


#fig.colorbar(im)
#fig.subplots_adjust(right=0.8)
#cbar_ax = fig.add_axes([0.85, 0.15, 0.05, 0.7])
#fig.colorbar(im, cax=cbar_ax)
```

```
## 3D plot, gradient
#X = np.unique(dataMatrix[:,0])
#Y = np.unique(dataMatrix[:,1])
#X, Y = np.meshgrid(X, Y)

#plt.figure()
#Vgy=np.empty([NumPosY,NumPosX,4])
#.....................................................
#Vgx=Vgy
#Vgx, Vgy=np.gradient(V10,X[1,1],Y[1,1])
```

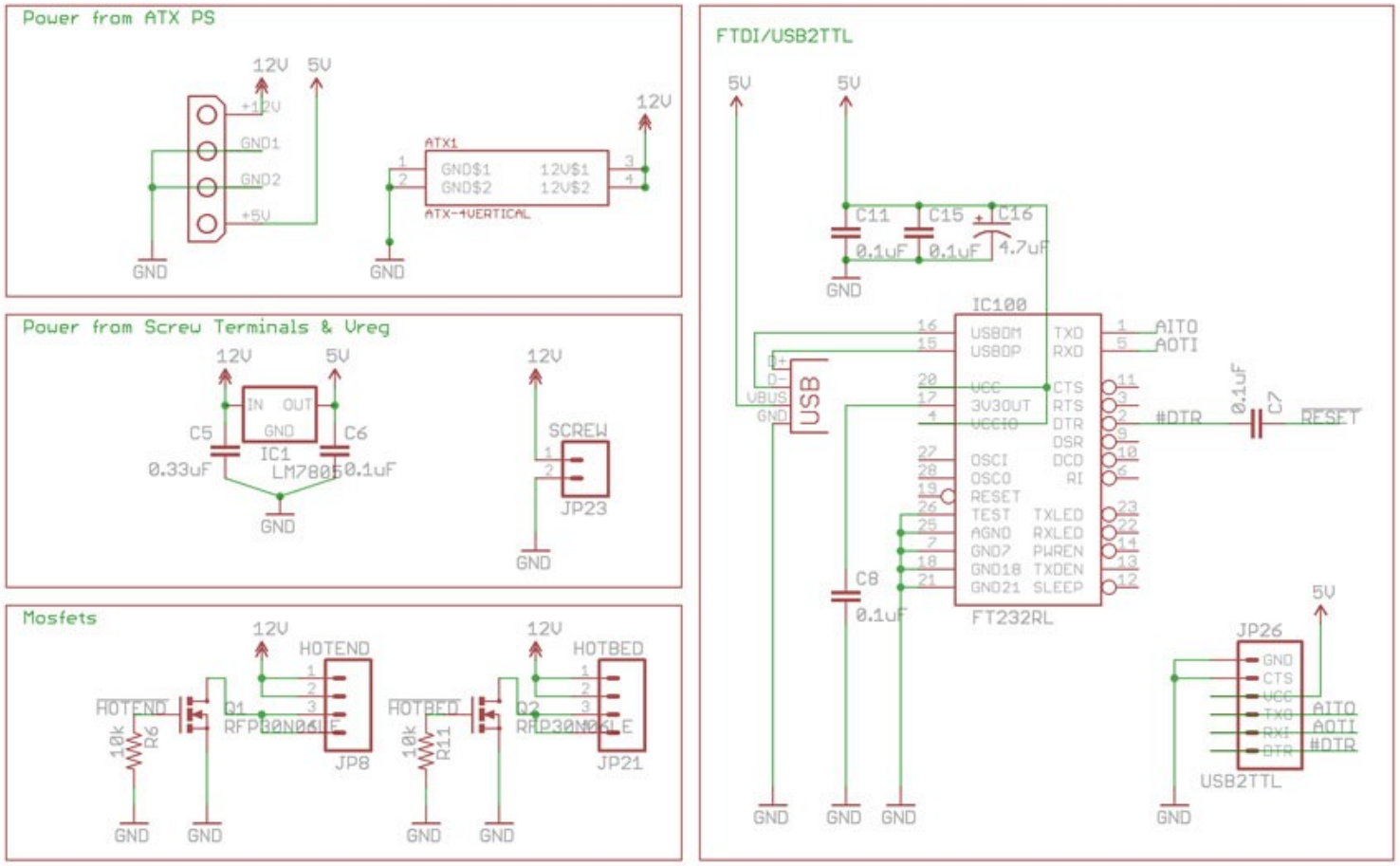
```
#plt.imshow(Vgy+Vgx,cmap=cm.coolwarm,origin='low',interpolation='neares
t')



#plt.imshow(Vg[:,:,0])

#ax = fig.gca(projection='3d')
#surf   =   ax.plot_surface(X,   Y,   np.flipud(V[:,:,0]),   rstride=1,
cstride=1, cmap=cm.coolwarm, linewidth=0, antialiased=False)
#ax.set_zlim(V[:,:,0].min(), V[:,:,0].max())
#ax.zaxis.set_major_locator(LinearLocator(10))
#ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
#fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()

sys.exit(0)
```

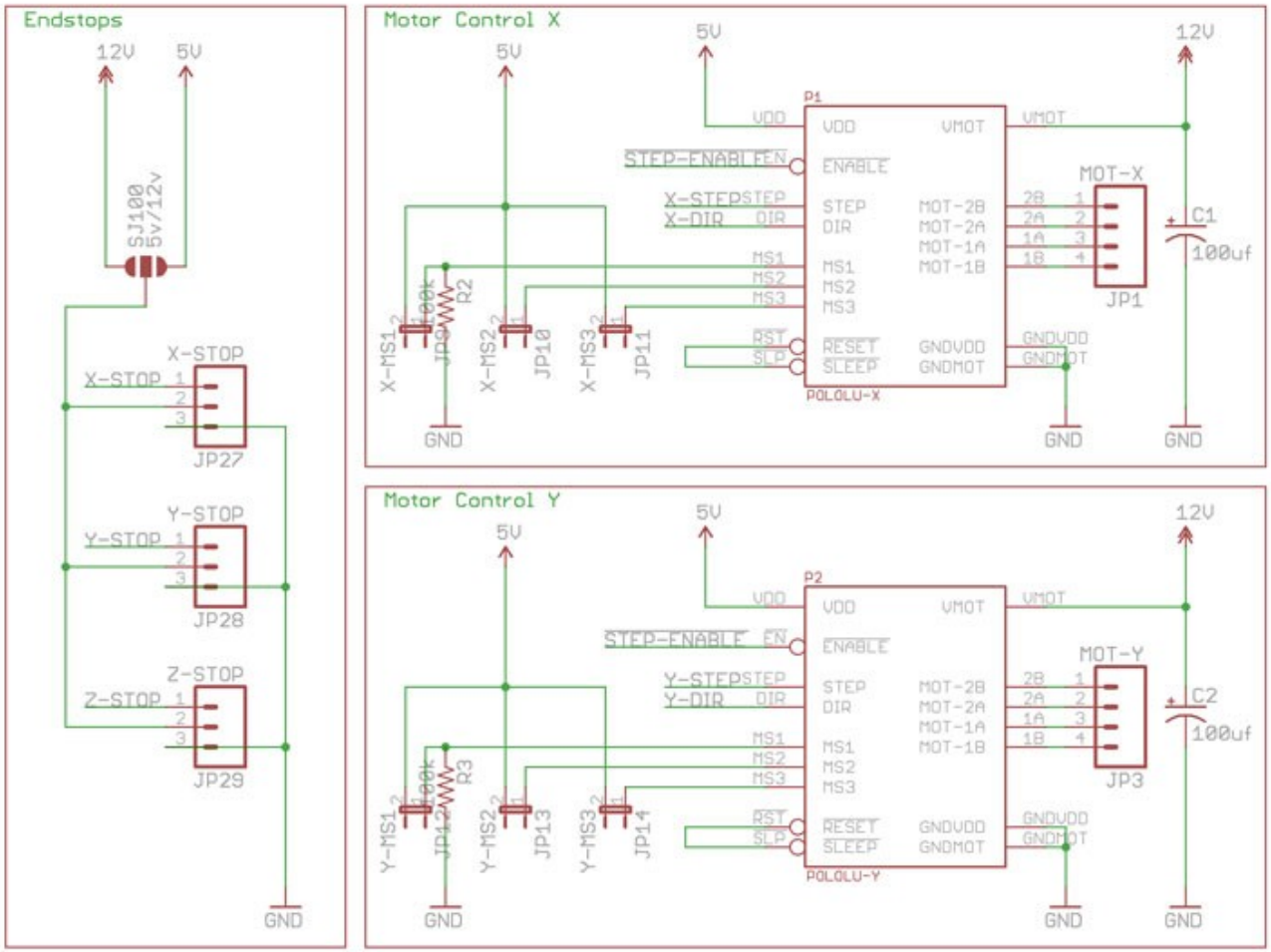
Histogram.py

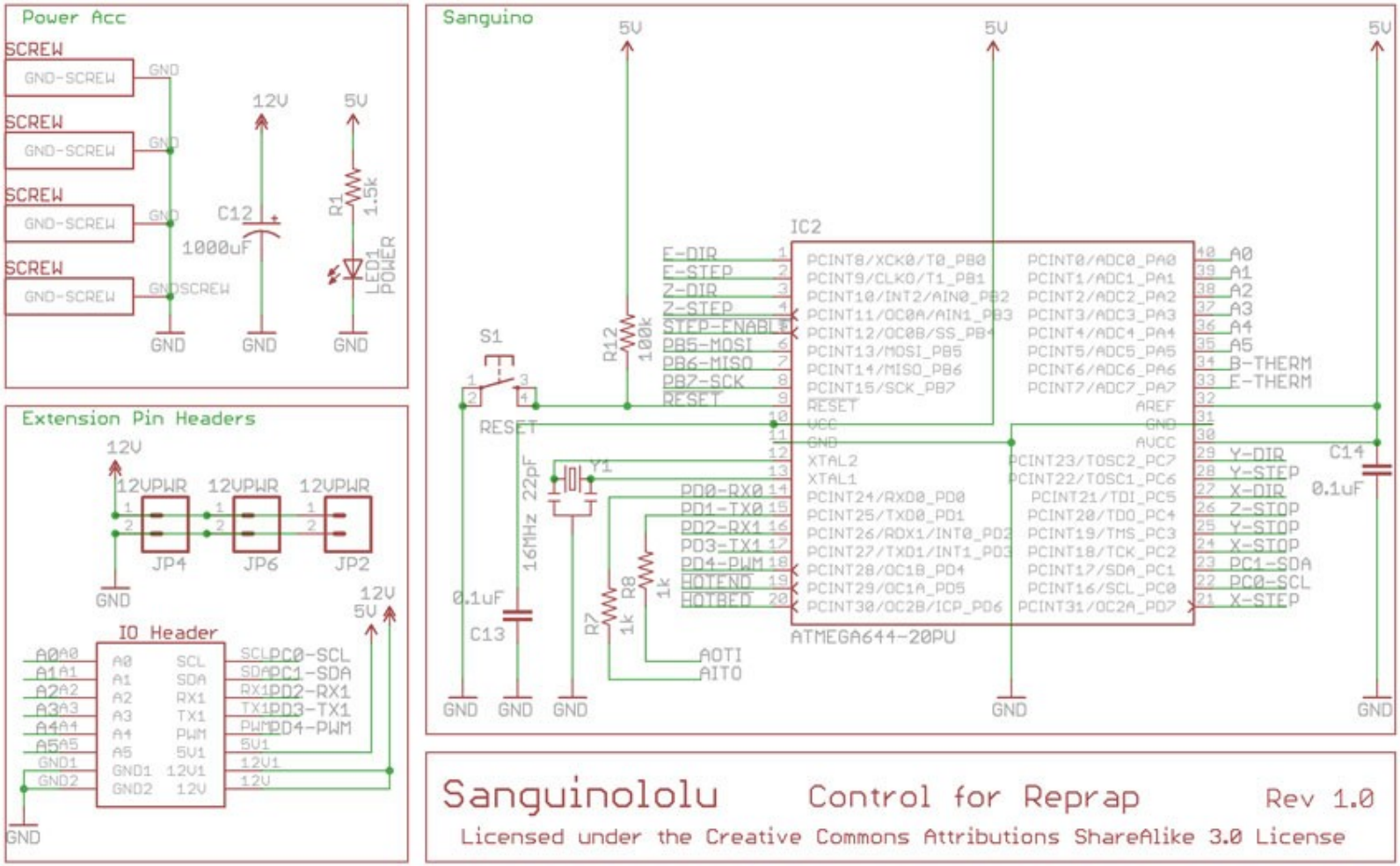
```
import matplotlib.pyplot as plt
import numpy as np
import sys

name = sys.argv[1]  # Name of csv file


try:
        fd = open(name,'r')    # Trying to open

except:
        print('No se pudo abrir el archivo indicado')
        sys.exit(0) # quit Python

dataMatrix=np.loadtxt(fd,delimiter=";",skiprows=1)
fd.close()

#Full data histogram
Values = dataMatrix[:,4] #dataMatrix[:,4->7]
numValues = len(np.unique(Values))
plt.hist(Values,numValues)
plt.title("Distribucion estadistica de las medidas")
plt.xlabel("Valor")
plt.ylabel("Frecuencia")

plt.show()


sizeMatrix=dataMatrix.shape #(rows, colums)


meanValue=np.empty([sizeMatrix[0], 4])

j=0
for i in range(sizeMatrix[0]):
        meanValue[j] = np.sum(dataMatrix[:i+1,4:], axis=0)/(i+1)
        j=j+1


meanValue -= meanValue[-1,:]


plt.plot(meanValue)
plt.title("Tendencia de las medias halladas")
plt.xlabel("Muestras")
plt.ylabel("Valor")
plt.legend(["V10","V20","V13","V23"])
```

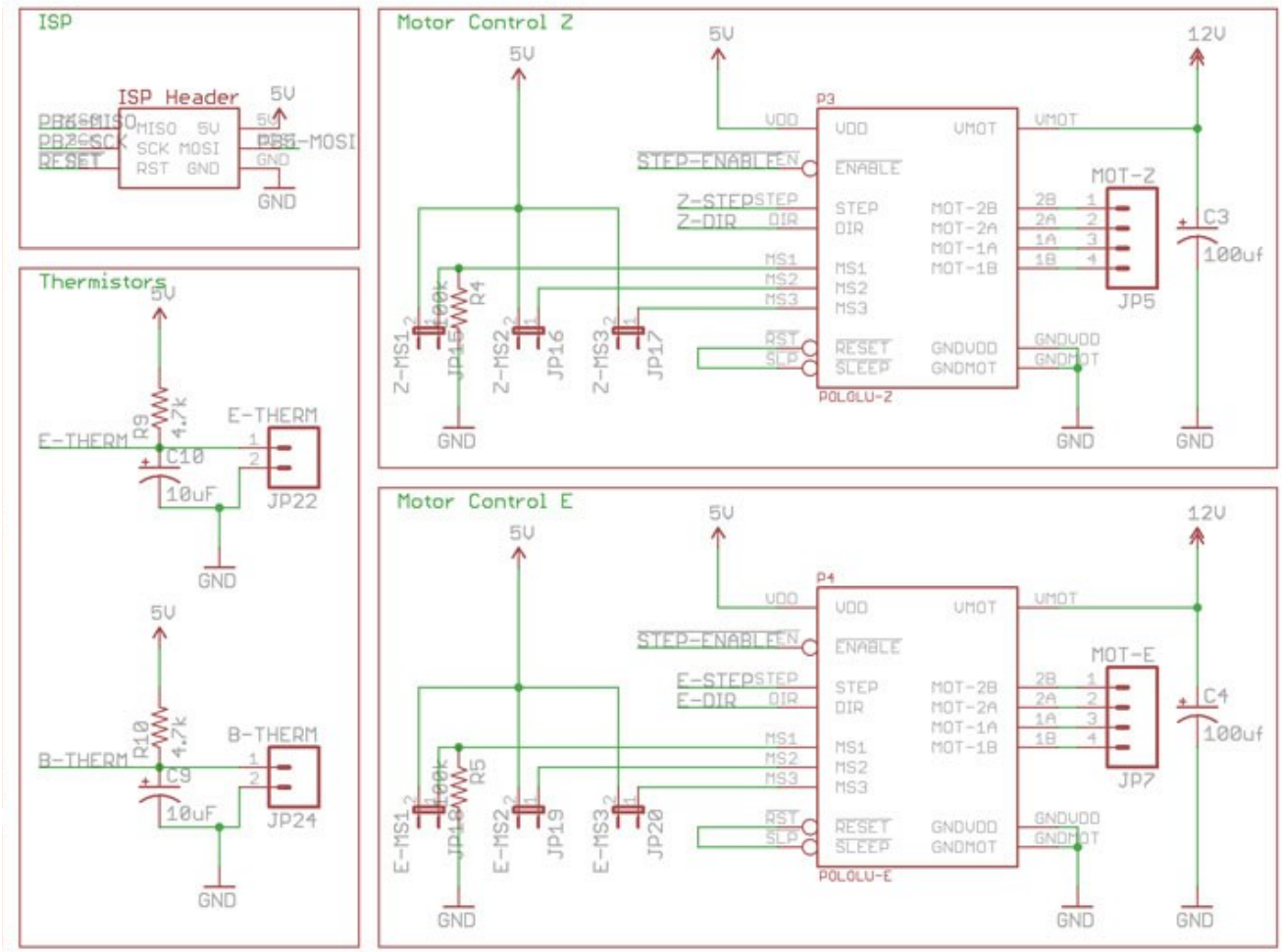
```
plt.show()
sys.exit(0)
```

H. Esquemático de la placa Sanguinololu




Endstops
12V
5V
SJ100
5v/12v
X-STOP
X-STOP
JP27
Y-STOP
Y-STOP
JP28
Z-STOP
Z-STOP
JP29
GND
Motor Control X
5V
12V
P1
VDD
VMOT
STEP-ENABLE
EN
ENABLE
X-STEP
STEP
X-DIR
DIR
MOT-2B
MOT-2A
MOT-1A
MOT-1B
MS1
MS2
MS3
RST
SLP
RESET
SLEEP
GNDVDD
GNDMOT
POLOLU-X
MOT-X
2B
2A
1A
1B
JP1
C1
100uf
X-MS1
X-MS2
X-MS3
100k
R2
JP9
JP10
JP11
Motor Control Y
P2
Y-STEP
Y-DIR
POLOLU-Y
MOT-Y
JP3
C2
Y-MS1
Y-MS2
Y-MS3
R3
JP12
JP13
JP14


Power Acc
SCREW
GND-SCREW
GND
SCREW
GND-SCREW
GND
SCREW
GND-SCREW
GND
SCREW
GND-SCREW
GNDSCREW
12V
5V
C12
1000uF
R1
1.5k
LED1
POWER
GND
GND
GND
Extension Pin Headers
12V
12VPWR
12VPWR
12VPWR
JP4
JP6
JP2
GND
IO Header
12V
5V
A0 A0
A1 A1
A2 A2
A3 A3
A4 A4
A5 A5
GND1
GND2
A0 SCL SCLPC0-SCL
A1 SDA SDAPC1-SDA
A2 RX1 RX1PD2-RX1
A3 TX1 TX1PD3-TX1
A4 PWM PWMPD4-PWM
A5 5V1 5V1
GND1 12V1 12V1
GND2 12V 12V
GND
Sanguino
5V
5V
5V
IC2
R12
100k
S1
RESET
E-DIR
E-STEP
Z-DIR
Z-STEP
STEP-ENABLE
PB5-MOSI
PB6-MISO
PB7-SCK
RESET
PCINT8/XCK0/T0_PB0
PCINT9/CLKO/T1_PB1
PCINT10/INT2/AIN0_PB2
PCINT11/OC0A/AIN1_PB3
PCINT12/OC0B/SS_PB4
PCINT13/MOSI_PB5
PCINT14/MISO_PB6
PCINT15/SCK_PB7
RESET
VCC
GND
XTAL2
XTAL1
PCINT24/RXD0_PD0
PCINT25/TXD0_PD1
PCINT26/RDX1/INT0_PD2
PCINT27/TXD1/INT1_PD3
PCINT28/OC1B_PD4
PCINT29/OC1A_PD5
PCINT30/OC2B/ICP_PD6
PCINT0/ADC0_PA0
PCINT1/ADC1_PA1
PCINT2/ADC2_PA2
PCINT3/ADC3_PA3
PCINT4/ADC4_PA4
PCINT5/ADC5_PA5
PCINT6/ADC6_PA6
PCINT7/ADC7_PA7
AREF
GND
AVCC
PCINT23/TOSC2_PC7
PCINT22/TOSC1_PC6
PCINT21/TDI_PC5
PCINT20/TDO_PC4
PCINT19/TMS_PC3
PCINT18/TCK_PC2
PCINT17/SDA_PC1
PCINT16/SCL_PC0
PCINT31/OC2A_PD7
ATMEGA644-20PU
A0
A1
A2
A3
A4
A5
B-THERM
E-THERM
Y-DIR
Y-STEP
X-DIR
Z-STOP
Y-STOP
X-STOP
PC1-SDA
PC0-SCL
X-STEP
C14
0.1uF
Y1
16MHz 22pF
0.1uF
C13
R7
1k
R8
1k
PD0-RX0
PD1-TX0
PD2-RX1
PD3-TX1
PD4-PWM
HOTEND
HOTBED
AOTI
AITO
GND
GND
GND
GND
GND
Sanguinololu
Control for Reprap
Rev 1.0
Licensed under the Creative Commons Attributions ShareAlike 3.0 License


ISP
ISP Header
5V
MISO 5V
SCK MOSI
RST GND
PB4-MISO
PB2-SCK
RESET
PB3-MOSI
GND
Thermistors
5V
R9
4.7k
E-THERM
C10
10uF
JP22
GND
R10
4.7k
B-THERM
C9
10uF
JP24
Motor Control Z
P3
VDD
VMOT
ENABLE
STEP-ENABLE
Z-STEP
Z-DIR
STEP
DIR
MS1
MS2
MS3
RST
SLP
RESET
SLEEP
MOT-2B
MOT-2A
MOT-1A
MOT-1B
GNDVDD
GNDMOT
POLOLU-2
MOT-Z
2B
2A
1A
1B
JP5
12V
C3
100uf
R4
100k
Z-MS1
Z-MS2
Z-MS3
JP15
JP16
JP17
Motor Control E
P4
E-STEP
E-DIR
E-MS1
E-MS2
E-MS3
R5
JP19
JP20
POLOLU-E
MOT-E
JP7
C4